Code Review Example(Authentication)

Here will we check how a whitebox pentestring approcach works over the example code in the same folder. You will need VSCODE & NODEJS installed on you clie

App.js

This part will spin up a new NODEJS server and it’s routes

// set up express

const app = express();

const port = parseInt("5000");

 // set up body parser and cors

app.use(bodyParser.json());

 // set up API routes

app.use("/api/auth", authRoutes);

app.use("/api/service", serviceRoutes);

 // start the Express server

app.listen(port, () => {

console.log(`⚡️[server]: Server is running at http://localhost:${port}`);

console.log(`⚡️[api]: APIs are running at <http://localhost:${port}/api`);>

});

The app.js was only the server setup side, now we want to check the authentication route and specifically if any of the fuctions used can be used to our needs.

const express = require("express");

const { getUserToken } = require("../controllers/auth-controllers.js");

const router = express.Router();

 router.post("/authenticate", getUserToken);

 module.exports = router;

In this case the api route is loading a internal function that deals to generate a JWT token so nothing interesting happens here.

service-routes.js

This route in other hand is loading another class used for string validation and QR and can be reached via api/service/generate:

const express = require("express");

const { verifyToken } = require("../controllers/auth-controllers.js");

const { generateQR } = require("../controllers/service-controllers.js");

const router = express.Router();

router.use(verifyToken);

router.post("/generate", generateQR);

module.exports = router;

Now here is where things get interesting here, there is a function that supposely vulnerable called “Validate String” as is used in tandem with Generate QR code for string validation(used the eval function) where basically only check for 3 chars.

function validateString(input, onError) {

if (

typeof input !== "string" ||

input.length == 0 ||

input.match(/['"`;]/g)

) {

eval(onError);

return false;

}

return true;

}

async function generateQR(req, res, next) {

const { text } = req.body;

const role = req.user.role;

try {

!validateString(

text,

// provide verbose error message 'for admins only'

role === "admin"

? `throw({message: 'The input "${text}" contains the following invalid characters: [${text.match(

/['"`;]/g

)}]', statusCode: 403})`

: "throw({message: 'Invalid input', statusCode: 403})"

);

As you see here the /generate function get’s from a HTTP POST request body the role as string and validates the role by calling that function...

Here it checks for:

* String must be, no numbers or other stuff
* Lenght more than 0
* The regex checks that ( “ ‘ ; ` ) aren’t present in the role