import React, { useState, useEffect } from 'react';

import { Button, Container, Typography, CircularProgress, Tabs, Tab, Box, Select, MenuItem, FormControl, InputLabel, Grid, Paper, AppBar, Toolbar } from '@mui/material';

import { Line } from 'react-chartjs-2';

import axios from 'axios';

import io from 'socket.io-client';

import { Chart, CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend } from 'chart.js';

import { createTheme, ThemeProvider } from '@mui/material/styles';

import { FaNetworkWired, FaDownload, FaLock, FaChartLine, FaCogs, FaTasks, FaFileAlt } from 'react-icons/fa';

import './App.css';

import AdminTools from './Admin';

// Register the necessary Chart.js components

Chart.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend);

const socket = io('http://localhost:5000');

const theme = createTheme({

palette: {

primary: {

main: '#0d47a1',

},

secondary: {

main: '#b71c1c',

},

},

typography: {

h4: {

fontWeight:'lighter',

color: '#fff',

},

h6: {

fontWeight: '500',

color:'lightblue',

},

},

components: {

MuiPaper: {

styleOverrides: {

root: {

padding: '20px',

marginTop: '20px',

borderRadius: '8px',

color:'black',

boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)',

backgroundColor: 'lightyellow',

},

},

},

MuiButton: {

styleOverrides: {

root: {

textTransform: 'none',

borderRadius: '8px',

},

},

},

MuiTabs: {

styleOverrides: {

root: {

backgroundColor:'lightblue',

color: '#fff',

marginBottom: '20px',

},

},

},

MuiTab: {

styleOverrides: {

root: {

textTransform: 'none',

fontWeight: 'bold',

fontSize: '16px',

},

},

},

MuiAppBar: {

styleOverrides: {

root: {

color:'purple',

backgroundColor: 'lightgreen',

marginBottom: '20px',

},

},

},

},

});

const App = () => {

const [loading, setLoading] = useState(false);

const [graphData, setGraphData] = useState({

labels: [],

datasets: [{

label: 'Packet Statistics',

data: [],

backgroundColor: ['#66b2ff', '#ff7043', '#ffd9b3']

}]

});

const [mitmDetails, setMitmDetails] = useState([]);

const [packets, setPackets] = useState([]);

const [capturing, setCapturing] = useState(false);

const [tabValue, setTabValue] = useState(0);

const [attackStats, setAttackStats] = useState({

mitm\_packets: 0,

spoofing\_packets: 0,

dns\_spoofing\_packets: 0,

https\_spoofing\_packets: 0,

total\_packets: 0

});

const [analysisData, setAnalysisData] = useState([]);

const [attackType, setAttackType] = useState('mitm\_attacks');

const [logs, setLogs] = useState('');

const [role, setRole] = useState('viewer'); // Default role is viewer

useEffect(() => {

socket.on('packet\_data', (data) => {

setPackets((prevPackets) => [...prevPackets, data]);

updateGraph(data);

});

return () => {

socket.off('packet\_data');

};

}, []);

const startCapture = async () => {

try {

setLoading(true);

setCapturing(true);

await axios.post('http://localhost:5000/start\_capture', { duration: 60 });

setLoading(false);

} catch (error) {

console.error('Error starting packet capture:', error);

alert('Failed to start packet capture. Please ensure the backend server is running.');

setLoading(false);

}

};

const stopCapture = async () => {

try {

setCapturing(false);

await axios.post('http://localhost:5000/stop\_capture');

} catch (error) {

console.error('Error stopping packet capture:', error);

}

};

const updateGraph = (data) => {

setGraphData((prevState) => {

const newLabels = [...prevState.labels, new Date().toLocaleTimeString()];

const newDataset = [...prevState.datasets[0].data, data.total\_packets];

return {

...prevState,

labels: newLabels,

datasets: [{

...prevState.datasets[0],

data: newDataset

}]

};

});

};

const fetchAttackStats = async () => {

try {

const response = await axios.get('http://localhost:5000/network\_attacks');

setAttackStats(response.data);

} catch (error) {

console.error('Error fetching attack statistics:', error);

}

};

const fetchAnalysisData = async () => {

try {

const response = await axios.get(http://localhost:5000/analyze\_attacks?type=${attackType});

setAnalysisData(response.data);

} catch (error) {

console.error('Error fetching analysis data:', error);

setAnalysisData([]); // Ensure it is an array on error

}

};

const fetchLogs = async () => {

try {

const response = await axios.get('http://localhost:5000/logs');

setLogs(response.data.logs);

} catch (error) {

console.error('Error fetching logs:', error);

}

};

const downloadReport = () => {

const csvContent = [

["Source", "Destination", "Details"],

...mitmDetails.map(detail => [detail.ip\_src, detail.ip\_dst, detail.details])

]

.map(e => e.join(","))

.join("\n");

const blob = new Blob([csvContent], { type: 'text/csv;charset=utf-8;' });

const url = URL.createObjectURL(blob);

const link = document.createElement('a');

link.setAttribute('href', url);

link.setAttribute('download', 'MITM\_Report.csv');

link.click();

};

const downloadCapturedPackets = () => {

window.location.href = 'http://localhost:5000/download\_packets';

};

const fetchThreatIntelligence = async () => {

try {

const response = await axios.get('http://localhost:5000/fetch\_threat\_intelligence');

console.log(response.data);

} catch (error) {

console.error('Error fetching threat intelligence:', error);

}

};

const handleChangeTab = (event, newValue) => {

setTabValue(newValue);

if (newValue === 3) {

fetchAttackStats();

} else if (newValue === 4) {

fetchAnalysisData();

} else if (newValue === 5) {

fetchLogs();

}

};

const handleChangeAttackType = (event) => {

setAttackType(event.target.value);

fetchAnalysisData();

};

const handleRoleChange = (event) => {

setRole(event.target.value);

};

const calculatePercentage = (part, total) => {

return total === 0 ? 0 : ((part / total) \* 100).toFixed(2);

};

return (

<ThemeProvider theme={theme}>

<Container>

<Typography variant="h4" gutterBottom>Network Monitor - {role.toUpperCase()}</Typography>

<FormControl variant="outlined" fullWidth style={{ marginBottom: '20px' }}>

<InputLabel>Role</InputLabel>

<Select value={role} onChange={handleRoleChange} label="Role">

<MenuItem value="admin">Admin</MenuItem>

<MenuItem value="viewer">Viewer</MenuItem>

</Select>

</FormControl>

<Tabs value={tabValue} onChange={handleChangeTab} centered>

<Tab icon={<FaNetworkWired />} label="Capture" />

<Tab icon={<FaChartLine />} label="Graph" />

<Tab icon={<FaDownload />} label="Download" />

<Tab icon={<FaLock />} label="Network Attacks" />

<Tab icon={<FaTasks />} label="Attack Analysis" />

<Tab icon={<FaFileAlt />} label="Logs" />

{role === 'admin' && <Tab icon={<FaCogs />} label="Admin Tools" />}

</Tabs>

{tabValue === 0 && (

<Grid container spacing={2}>

<Grid item>

<Button className="capture-button" variant="contained" onClick={startCapture} disabled={loading || capturing}>

{loading ? <CircularProgress size={24} /> : 'Start Packet Capture'}

</Button>

</Grid>

<Grid item>

<Button className="capture-button" variant="contained" onClick={stopCapture} disabled={!capturing}>

Stop Packet Capture

</Button>

</Grid>

</Grid>

)}

{tabValue === 1 && (

<Paper elevation={3} style={{ padding: '20px', marginTop: '20px' }}>

<div className="chart-container">

<Line data={graphData} />

</div>

</Paper>

)}

{tabValue === 2 && (

<Grid container spacing={2}>

<Grid item>

<Button className="download-button" variant="contained" onClick={downloadReport}>

Download MITM Report

</Button>

</Grid>

<Grid item>

<Button className="download-button" variant="contained" onClick={downloadCapturedPackets}>

Download Captured Packets

</Button>

</Grid>

</Grid>

)}

{tabValue === 3 && (

<Paper elevation={3} style={{ padding: '20px', marginTop: '20px' }}>

<Typography variant="h6">Network Attack Statistics</Typography>

<Typography variant="body1">Total Packets: {attackStats.total\_packets}</Typography>

<Typography variant="body1">MITM Packets: {attackStats.mitm\_packets} ({calculatePercentage(attackStats.mitm\_packets, attackStats.total\_packets)}%)</Typography>

<Typography variant="body1">Spoofing Packets: {attackStats.spoofing\_packets} ({calculatePercentage(attackStats.spoofing\_packets, attackStats.total\_packets)}%)</Typography>

<Typography variant="body1">DNS Spoofing Packets: {attackStats.dns\_spoofing\_packets} ({calculatePercentage(attackStats.dns\_spoofing\_packets, attackStats.total\_packets)}%)</Typography>

<Typography variant="body1">HTTPS Spoofing Packets: {attackStats.https\_spoofing\_packets} ({calculatePercentage(attackStats.https\_spoofing\_packets, attackStats.total\_packets)}%)</Typography>

</Paper>

)}

{tabValue === 4 && (

<Paper elevation={3} style={{ padding: '20px', marginTop: '20px' }}>

<Typography variant="h6">Attack Analysis</Typography>

<FormControl variant="outlined" fullWidth style={{ marginBottom: '20px' }}>

<InputLabel>Attack Type</InputLabel>

<Select value={attackType} onChange={handleChangeAttackType} label="Attack Type">

<MenuItem value="mitm\_attacks">MITM Attacks</MenuItem>

<MenuItem value="spoofing\_attacks">Spoofing Attacks</MenuItem>

<MenuItem value="dns\_spoofing\_attacks">DNS Spoofing Attacks</MenuItem>

<MenuItem value="https\_spoofing\_attacks">HTTPS Spoofing Attacks</MenuItem>

</Select>

</FormControl>

<pre className="logs-pre">

{JSON.stringify(analysisData, null, 2)}

</pre>

</Paper>

)}

{tabValue === 5 && (

<Paper elevation={3} style={{ padding: '20px', marginTop: '20px' }}>

<Typography variant="h6">Logs</Typography>

<pre className="logs-pre">

{logs}

</pre>

</Paper>

)}

{role === 'admin' && tabValue === 6 && (

<Paper elevation={3} style={{ padding: '20px', marginTop: '20px' }}>

<Typography variant="h6">Admin Tools</Typography>

<Button variant="contained" color="secondary" onClick={fetchThreatIntelligence}>

Fetch Threat Intelligence

</Button>

</Paper>

)}

</Container>

</ThemeProvider>

);

}