```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>AI-Driven Security Assistant</title>
<!--
AI-Driven Security Assistant
==========================
This is a single-file prototype built with HTML, CSS, and native JavaScript.
It simulates key functionalities for educational and demonstration purposes:
- Automatic phishing detection via keyword/URL analysis.
- Real-time network traffic monitoring with anomaly detection.
- Auto-generated incident response suggestions.
- Visual dashboard for security teams.

To run:
1. Save this as 'security_assistant.html'.
2. Open in VS Code (use Live Server extension for auto-reload, or open directly in browser).
3. Interact via the interface—no server required.

Limitations:
- Simulations only; no real ML or external integrations.
- For production, integrate with actual APIs (e.g., VirusTotal for phishing, SIEM for logs).
- Modular JS functions allow easy extension (e.g., replace simulations with real data fetches).

Author: Advanced Software Developer (Cybersecurity Focus)
Version: 1.0
-->
<style>
/* CSS Styling: Modern, professional, responsive design */
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}

body {
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #f4f7fa;
color: #333;
line-height: 1.6;
```

```
}

header {
background: linear-gradient(135deg, #1e3c72, #2a5298);
color: white;
padding: 1rem 2rem;
text-align: center;
box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

header h1 {
margin-bottom: 0.5rem;
}

main {
max-width: 1200px;
margin: 2rem auto;
padding: 0 1rem;
}

section {
background: white;
margin-bottom: 2rem;
padding: 1.5rem;
border-radius: 8px;
box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

h2 {
color: #1e3c72;
border-bottom: 2px solid #e0e6ed;
padding-bottom: 0.5rem;
margin-bottom: 1rem;
}

/ Phishing Detection /
#phishing-input {
width: 100%;
padding: 0.75rem;
border: 1px solid #ddd;
border-radius: 4px;
margin-bottom: 1rem;
```

```
resize: vertical;
min-height: 100px;
}

button {
background: #1e3c72;
color: white;
padding: 0.75rem 1.5rem;
border: none;
border-radius: 4px;
cursor: pointer;
font-size: 1rem;
transition: background 0.3s;
}

button:hover {
background: #2a5298;
}

.result {
margin-top: 1rem;
padding: 1rem;
border-radius: 4px;
font-weight: bold;
}

.phishing { background: #ffebee; color: #c62828; border-left: 4px solid #c62828; }
.legit { background: #e8f5e8; color: #2e7d32; border-left: 4px solid #2e7d32; }

/ Network Logs /
table {
width: 100%;
border-collapse: collapse;
margin-top: 1rem;
}

th, td {
padding: 0.75rem;
text-align: left;
border-bottom: 1px solid #ddd;
}
```

```
th {
background: #f8f9fa;
font-weight: bold;
}

.anomaly {
background: #ffebee;
color: #c62828;
}

/* Dashboard Metrics */
.metrics-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
gap: 1rem;
margin-top: 1rem;
}

.metric-card {
background: #f8f9fa;
padding: 1rem;
border-radius: 4px;
text-align: center;
}

.metric-value {
font-size: 2rem;
font-weight: bold;
color: #1e3c72;
}

/* Alerts */
.alert {
padding: 1rem;
margin: 1rem 0;
border-radius: 4px;
border-left: 4px solid;
}

.alert-high { background: #ffebee; color: #c62828; border-color: #c62828; }
.alert-medium { background: #fff3e0; color: #ef6c00; border-color: #ef6c00; }
.alert-low { background: #e8f5e8; color: #2e7d32; border-color: #2e7d32; }
```

```
/ Real-time Indicator /
#realtime-indicator {
position: fixed;
top: 1rem;
right: 1rem;
background: #4caf50;
color: white;
padding: 0.5rem 1rem;
border-radius: 20px;
font-size: 0.9rem;
display: none;
}

/ Responsive Design /
@media (max-width: 768px) {
main {
padding: 0 0.5rem;
margin: 1rem auto;
}

section {
padding: 1rem;
}

table {
font-size: 0.9rem;
}

th, td {
padding: 0.5rem;
}

.metrics-grid {
grid-template-columns: 1fr;
}
}

/ Simple Progress Bar for Anomalies /
.progress-bar {
width: 100%;
height: 20px;
```

```
background: #ddd;
border-radius: 10px;
overflow: hidden;
margin: 0.5rem 0;
}

.progress-fill {
height: 100%;
background: #1e3c72;
transition: width 0.3s;
}
</style>
</head>
<body>
<header>
<h1>AI-Driven Security Assistant</h1>
<p>Automated Detection, Monitoring, and Response for Evolving Cyber Threats</p>
</header>

<div id="realtime-indicator">Monitoring Active</div>

<main>
<!-- Dashboard Overview Section -->
<section id="dashboard">
<h2>Visual Dashboard</h2>
<p>Real-time overview of security metrics and threats.</p>
<div class="metrics-grid" id="metrics-grid">
<!-- Metrics will be populated by JS -->
</div>
<div class="progress-bar">
<div class="progress-fill" id="anomaly-progress" style="width: 0%;"></div>
</div>
<p id="anomaly-status">Anomaly Rate: 0%</p>
</section>

<!-- Phishing Detection Section -->
<section id="phishing-detection">
<h2>1. Automatic Phishing Detection</h2>
<p>Enter email content or URL to analyze for phishing indicators (e.g., suspicious keywords, domains).</p>
<textarea id="phishing-input" placeholder="Example: Urgent! Click http://fake-bank.com to update password..."></textarea>
```

```
<button onclick="detectPhishing()">Detect Phishing</button>
<div id="phishing-result"></div>
</section>

<!-- Network Monitoring Section -->
<section id="network-monitoring">
<h2>2. Real-Time Network Traffic Monitoring</h2>
<p>Simulated logs of network activity. Anomalies are flagged based on traffic volume and suspicious patterns.</p>
<button onclick="generateNewLog()">Generate New Log Entry</button>
<table id="network-logs">
<thead>
<tr>
<th>Timestamp</th>
<th>Source IP</th>
<th>Destination</th>
<th>Bytes</th>
<th>Port</th>
<th>Anomaly Score</th>
</tr>
</thead>
<tbody>
<!-- Logs populated by JS -->
</tbody>
</table>
</section>

<!-- Incident Response Section -->
<section id="incident-response">
<h2>3. Auto-Generated Incident Response Suggestions</h2>
<p>Suggestions based on detected threats. Severity levels guide actions.</p>
<div id="response-suggestions"></div>
</section>
</main>

<script>
// JavaScript: Modular Functions for AI-Driven Security Assistant
// All logic is native JS; simulations mimic real behaviors.
// Modules: Phishing Detection, Network Monitoring, Incident Response, Dashboard.

// Global State
let networkLogs = []; // Array of log objects
```

```
let phishingDetected = false;
let anomalyCount = 0;
let totalLogs = 0;

// Module 1: Phishing Detection
/**
* Detects phishing by analyzing input for suspicious patterns.
* Simulation: Checks for keywords (e.g., 'urgent', 'click'), fake domains, etc.
* In production: Integrate with NLP APIs or ML models via fetch().
* @param {string} input - Email/URL content to analyze.
* @returns {Object} - Detection result with confidence score.
*/
function analyzePhishing(input) {
const suspiciousKeywords = ['urgent', 'click here', 'update password', 'winner', 'prize', 'verify
now'];
const suspiciousDomains = ['fake', 'scam', 'phish', 'suspicious'];
const score = suspiciousKeywords.filter(word => input.toLowerCase().includes(word)).length +
suspiciousDomains.filter(domain => input.toLowerCase().includes(domain)).length;
const confidence = Math.min(score * 20, 100); // Simulated confidence (0-100%)
const isPhishing = score > 1; // Threshold for detection
return { isPhishing, confidence };
}

/**
* Handles phishing detection UI and updates global state.
*/
function detectPhishing() {
const input = document.getElementById('phishing-input').value.trim();
if (!input) {
alert('Please enter content to analyze.');
return;
}

const result = analyzePhishing(input);
phishingDetected = result.isPhishing;

const resultDiv = document.getElementById('phishing-result');
const className = result.isPhishing ? 'phishing' : 'legit';
const message = result.isPhishing ? 'Phishing Detected!' : 'Legitimate Content';
resultDiv.innerHTML = <div class="result ${className}">${message}<br>Confidence:
${result.confidence.toFixed(1)}%</br></div>;
```

```
updateIncidentResponse();
updateDashboard();
}

// Module 2: Network Traffic Monitoring
/**
* Generates a simulated network log entry.
* Simulation: Random IPs, bytes, ports; flags anomalies (e.g., high bytes >5000, suspicious port 4444).
* In production: Fetch from logs API or WebSocket for real-time.
* @returns {Object} - Log entry.
*/
function generateLogEntry() {
const timestamp = new Date().toLocaleString();
const srcIP ='192.168.1.${Math.floor(Math.random() * 255) + 1}';
const dstIP = Math.random() > 0.7 ? 'suspicious-attacker.com'
:'10.0.0.${Math.floor(Math.random() * 255) + 1}';
const bytes = Math.floor(Math.random() * 9000) + 100;
const port = Math.random() > 0.8 ? 4444 : Math.floor(Math.random() * 400) + 80; // 80-480, 4444 suspicious
const anomalyScore = (bytes / 100) + (dstIP.includes('suspicious')? 50 : 0) + (port === 4444 ? 30 : 0);
const isAnomaly = anomalyScore > 50;

if (isAnomaly) anomalyCount++;

return { timestamp, srcIP, dstIP, bytes, port, anomalyScore, isAnomaly };
}

/**
* Adds a new log entry and updates the table.
*/
function generateNewLog() {
const log = generateLogEntry();
networkLogs.unshift(log); // Add to front for recent-first
totalLogs++;
renderNetworkLogs();
updateIncidentResponse();
updateDashboard();
}

/**
```

```
* Renders network logs in the table, highlighting anomalies.
*/
function renderNetworkLogs() {
const tbody = document.querySelector('#network-logs tbody');
tbody.innerHTML = '';
networkLogs.slice(0, 10).forEach(log => { // Show last 10
const row = tbody.insertRow();
row.className = log.isAnomaly ? 'anomaly' : '';
row.innerHTML = `
<td>${log.timestamp}</td>
<td>${log.srcIP}</td>
<td>${log.dstIP}</td>
<td>${log.bytes}</td>
<td>${log.port}</td>
<td>${log.anomalyScore.toFixed(1)}</td>
`;
});
}

// Module 3: Incident Response Generation
/**
* Generates response suggestions based on detections.
* Simulation: Rule-based logic for severity and actions.
* In production: Use decision trees or AI APIs for dynamic responses.
*/
function generateResponses() {
const suggestions = [];
let severity = 'Low';

if (phishingDetected) {
severity = 'Medium';
suggestions.push('Quarantine affected email/endpoint.', 'Block sender domain/URL in
firewall.', 'Alert users and team via notification system.');
}

if (anomalyCount > 0) {
severity = anomalyCount > 2 ? 'High' : 'Medium';
suggestions.push('Block suspicious IP/port via firewall rules.', 'Investigate with packet capture
tools.', 'Quarantine source devices from network.');
}

return { severity, suggestions };
```

```
}

/**
* Updates the incident response UI.
*/
function updateIncidentResponse() {
const { severity, suggestions } = generateResponses();
const div = document.getElementById('response-suggestions');
const alertClass = severity === 'High' ? 'alert-high' : severity === 'Medium' ? 'alert-medium' :
'alert-low';
div.innerHTML = `
<div class="alert ${alertClass}">
<strong>Incident Severity: ${severity}</strong><br>
${suggestions.map(sug => <li>${sug}</li>).join('')}
</div>
`;
}

// Module 4: Visual Dashboard
/**
* Updates dashboard metrics and visuals.
*/
function updateDashboard() {
const anomalyRate = totalLogs > 0 ? (anomalyCount / totalLogs * 100).toFixed(1) : 0;
document.getElementById('anomaly-progress').style.width = '${anomalyRate}%';
document.getElementById('anomaly-status').textContent = 'Anomaly Rate: ${anomalyRate}%';

const metricsGrid = document.getElementById('metrics-grid');
metricsGrid.innerHTML = `
<div class="metric-card">
<div class="metric-value">${totalLogs}</div>
<div>Total Logs</div>
</div>
<div class="metric-card">
<div class="metric-value">${phishingDetected ? '1' : '0'}</div>
<div>Phishing Alerts</div>
</div>
<div class="metric-card">
<div class="metric-value">${anomalyCount}</div>
<div>Anomalies</div>
</div>
<div class="metric-card">
```

```
<div class="metric-value"><1s</div>
<div>Response Time</div>
</div>
`;
}

// Initialization and Real-Time Simulation
/**
* Initializes the app: Generates initial logs, starts real-time monitoring.
*/
function init() {
// Generate initial logs
for (let i = 0; i < 5; i++) {
generateNewLog();
}
updateIncidentResponse();
updateDashboard();

// Simulate real-time: Add log every 5 seconds
setInterval(() => {
generateNewLog();
document.getElementById('realtime-indicator').style.display= 'block';
}, 5000);

// Event listeners for responsiveness
window.addEventListener('resize', updateDashboard);
}

// Start the app when DOM is loaded
document.addEventListener('DOMContentLoaded', init);
</script>
</body>
</html>
```