PHISHING SIMULATION

FEATURES:

1. Admin Dashboard

- **User Management**: Manage employee profiles and group them for targeted campaigns.
- **Campaign Creation**: Design and schedule phishing simulations using customizable templates.
- **Reporting**: Generate reports on user interactions, including click rates and data submissions.

2. Phishing Email Templates

- **Template Library**: Pre-designed templates that mimic common phishing scenarios.
- **Customization Options**: Allow administrators to edit content to suit organizational needs.

3. Email Delivery System

- **Bulk Sending**: Dispatch emails to selected user groups.
- **Tracking**: Embed tracking links to monitor user actions.

4. User Interaction Tracking

- **Link Clicks**: Identify users who click on phishing links.
- **Form Submissions**: Track data entered into simulated phishing forms.

5. Educational Resources

- **Immediate Feedback**: Inform users who interact with phishing elements about what they missed.
- **Training Materials**: Provide resources to help users recognize phishing attempts in the future.

Prerequisites

1. **Node.js**: install on your machine.
2. **Express.js**: Use Express.js to set up a simple server.
3. **Nodemailer**: For sending emails.
4. **MongoDB**: To store user interactions and feedback (you can use other databases like MySQL or PostgreSQL).

Step-by-Step Implementation

1. **Set Up the Server**
   - Create a new directory for your project and initialize it with npm
   - Run the following command in bash shell:

   mkdir phishing-simulation

   cd phishing-simulation

   npm init –y

   - Install the necessary packages in bash shell:

     npm install express nodemailer mongoose body-parser

   - Create an index.js file for your server logic:

Figure 1. JavaScript

```
// index.js
const express = require('express');
const nodemailer = require('nodemailer');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/phishing_simulation', { useNewUrlParser: true, useUnifiedTopology: true });

// Define a schema and model for user interactions
const interactionSchema = new mongoose.Schema({
  email: String,
  action: String, // e.g., "clicked link", "submitted form"
  timestamp: Date,
});

const Interaction = mongoose.model('Interaction', interactionSchema);

// Configure Nodemailer
const transporter = nodemailer.createTransport({
  service: 'Gmail', // You can use other services
  auth: {
    user: 'your-email@gmail.com',
    pass: 'your-email-password',
  },
});

// Endpoint to send phishing emails
app.post('/send-email', async (req, res) => {
  const { recipientEmail } = req.body;
  const mailOptions = {
    from: 'your-email@gmail.com',
    to: recipientEmail,
    subject: 'Security Alert: Verify Your Account',
    html: `
      <p>Dear User,</p>
      <p>We have detected unusual activity on your account. Please <a href="http://localhost:3000/track-click?email=${recipientEmail}">click here</a> to verify your account.</p>
    `,
  };
```

Figure 2. JavaScript

```
  try {
    await transporter.sendMail(mailOptions);
    res.send('Email sent successfully.');
  } catch (error) {
    res.status(500).send('Error sending email.');
  }
});

// Endpoint to track clicks
app.get('/track-click', async (req, res) => {
  const { email } = req.query;

  const interaction = new Interaction({
    email,
    action: 'clicked link',
    timestamp: new Date(),
  });

  await interaction.save();
  res.send('Click tracked. Thank you for your interaction.');
});

// Endpoint to get interactions (for admin/reporting)
app.get('/interactions', async (req, res) => {
  const interactions = await Interaction.find();
  res.json(interactions);
});

// Start the server
app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

2. Deploy Phishing Email Templates

- HTML template sent through the email:

Figure 3.HTML

```
<p>Dear User,</p>
<p>We have detected unusual activity on your account. Please <a href="http://localhost:3000/track-click?email=recipient@example.com">click here</a> to verify your account.</p>
```

- Replace recipient@example.com with dynamic values in your server logic to personalize emails

3. Track User Interactions

- When users click the link in the email, the /track-click endpoint logs their action to MongoDB.

4. Provide Educational Resources

- Create a basic HTML page for training:

Figure 4. HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Phishing Awareness Training</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .content { max-width: 600px; margin: auto; padding: 20px; }
    </style>
</head>
<body>
    <div class="content">
        <h1>Phishing Awareness Training</h1>
        <p>Thank you for participating in our phishing simulation. Here are some tips to help you identify phishing emails:</p>
        <ul>
            <li>Check the sender's email address carefully.</li>
            <li>Look for spelling and grammar mistakes.</li>
            <li>Be cautious of urgent requests for personal information.</li>
            <li>Hover over links to see the actual URL before clicking.</li>
        </ul>
        <p>Stay safe online!</p>
    </div>
</body>
</html>
```

By following these steps, you can create a functional and educational phishing simulation platform that helps organizations improve their cybersecurity posture and employee awareness.