DOCUMENTATION

How To Implement Server Side Validation

(One Line Draw CAPTCHA)

Introduction

The "Draw the Pattern" CAPTCHA validates users by having them draw a pattern on a canvas. However, to enhance security, server-side validation is required to prevent bots from bypassing the CAPTCHA by manipulating client-side scripts.

2. Server Validation Flow

1. The client draws a pattern according to generated dots.
2. Pattern coordinate data is sent to the server for validation.
3. The server evaluates the order and accuracy of the pattern against predefined dots.
4. If valid, the server returns a success response. If not, authentication is denied.

Steps to Implement Server-Side CAPTCHA Validation (Example using NodeJS)

1. Data Structure Sent by the Client

```
{
    "userPattern": [
        {"x": 50, "y": 100},
        {"x": 120, "y": 180},
        {"x": 200, "y": 220},
        {"x": 300, "y": 150}
    ]
}
```

2. Backend Implementation (Node.js with Express)

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');

app.use(bodyParser.json());

// Store the correct pattern on the server
const correctPattern = [
    { x: 50, y: 100 },
    { x: 120, y: 180 },
    { x: 200, y: 220 },
    { x: 300, y: 150 }
];

// Function to validate the pattern
function validatePattern(userPattern) {
    if (userPattern.length !== correctPattern.length) return
false;
    for (let i = 0; i < correctPattern.length; i++) {
        const userPoint = userPattern[i];
        const correctPoint = correctPattern[i];
        if (Math.abs(userPoint.x - correctPoint.x) > 10 ||
Math.abs(userPoint.y - correctPoint.y) > 10) {
            return false;
        }
    }
    return true;
}

// Validation endpoint
app.post('/validate-captcha', (req, res) => {
    const { userPattern } = req.body;
    if (!userPattern) {
        return res.status(400).json({ success: false, message:
"Invalid data format" });
    }

    if (validatePattern(userPattern)) {
        res.json({ success: true, message: "CAPTCHA validated
successfully" });
    } else {
        res.json({ success: false, message: "CAPTCHA validation
failed" });
    }
});

app.listen(3000, () => {
    console.log('Server running on port 3000');
     });
```

3. Client-Side Integration (JavaScript)

```
fetch('http://localhost:3000/validate-captcha', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({ userPattern: drawnPattern })
})
.then(response => response.json())
.then(data => {
    if (data.success) {
        alert('CAPTCHA passed!');
        window.location.href =
"https://mathewsin.github.io/CaptchaTester/";
    } else {
        alert('CAPTCHA failed, please try again.');
    }
})
.catch(error => console.error('Error:', error));
```

Conclusion

By implementing server-side validation, the CAPTCHA is more secure against manipulation. Ensure the server strictly verifies the received pattern and use HTTPS to prevent Man-In-The-Middle (MITM) attacks.