Armen Melkumyan • 1st
Technical / Solutions Architect
11mo •

🚀 Today I would like to share CHAP (Challenge-Handshake Authentication Protocol) for WebSockets!

🔐✨

WebSocket communication is essential for real-time applications, but securing it is paramount. Let's explore how CHAP can enhance your WebSocket security. 📡🔒

What is CHAP?

CHAP is an authentication scheme that uses a three-way handshake to verify the identity of the client. It’s a robust method to ensure that the communication is secure and authenticated. 💡

How Does CHAP Work?

1) Establish WebSocket Connection: The client initiates a connection to the server.

2) Server Sends Challenge: The server sends a unique challenge (nonce) to the client.

3) Client Responds: The client computes a response using a shared secret and the nonce, then sends it back to the server.

4) Server Verifies: The server verifies the client's response. If it matches, the client is authenticated.

The server generates and sends a nonce (challenge) to the client.

The client responds with a hash of the nonce and a shared secret.

The server verifies the response to authenticate the client.

Benefits of Using CHAP:

1) Security: Reduces the risk of replay attacks since each challenge is unique.

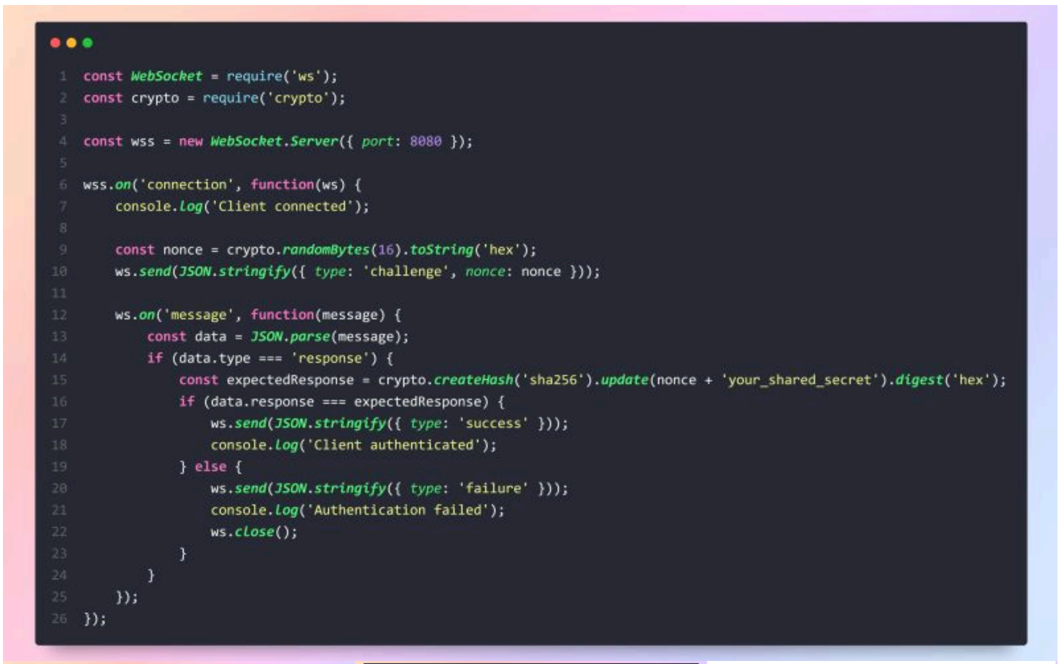
2)Flexibility: Can be implemented in various environments and programming languages.

3) Reliability: Periodically verifies the client during the session, ensuring continuous authentication.

🛡️ Implementing CHAP authentication enhances your WebSocket security, providing a reliable and flexible approach to ensure secure real-time communication.

In the attached images are implementation with JavaScript Node.js and Dot Net C#

#WebDevelopment #WebSockets #Security #Authentication #NodeJS #JavaScript #Coding #Tech #dotnet #Charp



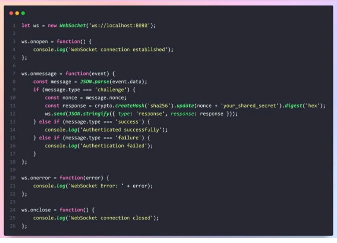
```javascript
const WebSocket = require('ws');
const crypto = require('crypto');

const wss = new WebSocket.Server({ port: 8080 });

wss.on('connection', function(ws) {
    console.log('Client connected');

    const nonce = crypto.randomBytes(16).toString('hex');
    ws.send(JSON.stringify({ type: 'challenge', nonce: nonce }));

    ws.on('message', function(message) {
        const data = JSON.parse(message);
        if (data.type === 'response') {
            const expectedResponse = crypto.createHash('sha256').update(nonce + 'your_shared_secret').digest('hex');
            if (data.response === expectedResponse) {
                ws.send(JSON.stringify({ type: 'success' }));
                console.log('Client authenticated');
            } else {
                ws.send(JSON.stringify({ type: 'failure' }));
                console.log('Authentication failed');
                ws.close();
            }
        }
    });
});
```



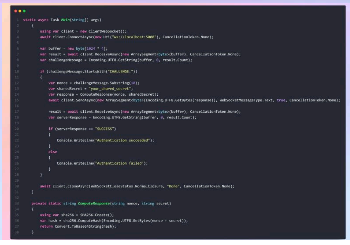
```javascript
let ws = new WebSocket('ws://localhost:8080');

ws.onopen = function() {
    console.log('WebSocket connection established');
};

ws.onmessage = function(event) {
    const message = JSON.parse(event.data);
    if (message.type === 'challenge') {
        const nonce = message.nonce;
        const response = crypto.createHash('sha256').update(nonce + 'your_shared_secret').digest('hex');
        ws.send(JSON.stringify({ type: 'response', response: response }));
    } else if (message.type === 'success') {
        console.log('Authenticated successfully');
    } else if (message.type === 'failure') {
        console.log('Authentication failed');
    }
};

ws.onerror = function(error) {
    console.log('WebSocket Error: ' + error);
};

ws.onclose = function() {
    console.log('WebSocket connection closed');
};
```



```csharp
static async Task Main(string[] args)
    {
        using var client = new ClientWebSocket();
        await client.ConnectAsync(new Uri("ws://localhost:5000"), CancellationToken.None);

        var buffer = new byte[1024 * 4];
        var result = await client.ReceiveAsync(new ArraySegment<byte>(buffer), CancellationToken.None);
        var challengeMessage = Encoding.UTF8.GetString(buffer, 0, result.Count);

        if (challengeMessage.StartsWith("CHALLENGE:"))
        {
            var nonce = challengeMessage.Substring(10);
            var sharedSecret = "your_shared_secret";
            var response = ComputeResponse(nonce, sharedSecret);
            await client.SendAsync(new ArraySegment<byte>(Encoding.UTF8.GetBytes(response)), WebSocketMessageType.Text, true, CancellationToken.None);

            result = await client.ReceiveAsync(new ArraySegment<byte>(buffer), CancellationToken.None);
            var serverResponse = Encoding.UTF8.GetString(buffer, 0, result.Count);

            if (serverResponse == "SUCCESS")
            {
                Console.WriteLine("Authentication succeeded");
            }
            else
            {
                Console.WriteLine("Authentication failed");
            }
        }

        await client.CloseAsync(WebSocketCloseStatus.NormalClosure, "Done", CancellationToken.None);
    }

    private static string ComputeResponse(string nonce, string secret)
    {
        using var sha256 = SHA256.Create();
        var hash = sha256.ComputeHash(Encoding.UTF8.GetBytes(nonce + secret));
        return Convert.ToBase64String(hash);
    }
```

+1

68

2 comments 3 reposts