Izvještaj laboratorijskih vježbi

Rato Kuzmanić, 250

Vježba: 4. CTR mode

Grupa: Grupa 2

Rješenje: Saddam Hussein was not found hiding in a “hole.” Saddam was roundhouse-kicked in the head by Chuck Norris in Kansas, which sent him through the earth, stopping just short of the surface of Iraq.

client.js

```
const http = require('http');
const xor = require('buffer-xor');
const { prettyLogSuccess, prettyLogError } = require('./logger');
const { app, request: { get: getRequest, post: postRequest } } =
require('./config');

getChallenge = () =>
    new Promise((resolve, reject) => {
        const request = http.request(getRequest, response => {
            let data = '';
            response.on('data', chunk => data += chunk);
            response.on('end', () => resolve(JSON.parse(data)));
        });
        request.end();
    });

getCiphertext = plaintext =>
    new Promise((resolve, reject) => {
        const data = JSON.stringify({ plaintext });

        const request = http.request(postRequest, response => {
            response.setEncoding('utf8');

            response.on('data', data => resolve(JSON.parse(data)));
            response.on('error', error => reject(error));
        });

        request.write(data);
        request.end();
    });

getZeroHexOfSameSizeAs = source => '0'.repeat(source.length);

isHit = possiblePlaintext => possiblePlaintext.includes('Chuck');

async function shouldAlwaysHaltAction(challengeCiphertext, payload) {
    let plaintext = null;

    for(let iteration = 0; iteration < app.maxIterationCount; iteration++) {
        let { ciphertext } = await getCiphertext(payload);
        let possiblePlaintext = xor(Buffer.from(challengeCiphertext, 'hex'),
Buffer.from(ciphertext, 'hex')).toString('utf8');
        if(isHit(possiblePlaintext)) {
            plaintext = possiblePlaintext;
            break;
        }
    }

    plaintext
    ? prettyLogSuccess('Joke found', plaintext)
    : prettyLogError('Joke not found', 'Maximum iteration count was reached. Try
increasing the threshold in config.js');
}

async function shouldntAlwaysHaltAction(challengeCiphertext, payload) {
    while(true) {
        let { ciphertext } = await getCiphertext(payload);
        let possiblePlaintext = xor(Buffer.from(challengeCiphertext, 'hex'),
Buffer.from(ciphertext, 'hex')).toString('utf8');
        if(isHit(possiblePlaintext)) {
            prettyLogSuccess('Joke found', possiblePlaintext);
            break;
        }
    }
}

(async () => {
    const { ciphertext } = await getChallenge();
    const payload = getZeroHexOfSameSizeAs(ciphertext);

    app.shouldAlwaysHalt
    ? shouldAlwaysHaltAction(ciphertext, payload)
    : shouldntAlwaysHaltAction(ciphertext, payload);
})();
```

logger.js

```
const chalk = require('chalk');

String.prototype.addWhitespacePadding = function(numberOfWhitespaces = 8) {
    return `${' '.repeat(numberOfWhitespaces)}${this}${'
'.repeat(numberOfWhitespaces)}`;
}
```

```
logError = (title, error) => {
    console.log(`\n${chalk.white.bgRed(title.addWhitespacePadding())}`);
    console.log(`Details: ${error}\n`);
}

logSuccess = (title, details) => {
    console.log(`\n${chalk.black.bgGreen(title.addWhitespacePadding())}`);
    console.log(`Details: ${details}\n`);
}

module.exports = {
    prettyLogError: logError,
    prettyLogSuccess: logSuccess
}
```

config.js

```
const app = {
    shouldAlwaysHalt: false,
    maxIterationCount: 5000
}

const commonRequest = {
    host: '10.0.0.6',
    port: 80,
    headers: {
        'Content-Type': 'application/json'
    }
};

const getRequest = {
    ...commonRequest,
    path: '/ctr/challenge',
    method: 'GET'
};

const postRequest = {
    ...commonRequest,
    path: '/ctr',
    method: 'POST'
};
```

```
module.exports = {
    app: app,
    request: {
        get: getRequest,
        post: postRequest
    }
}
```