Point-of-Sale Password Recovery

In this objective, we'll be pulling apart an application to find a hardcoded password. To get the hints, complete the **Linux Primer** terminal first.

Objective

> Help Sugarplum Mary in the Courtyard find the supervisor password for the point-of-sale terminal. What's the password?

`Difficulty: 1/5`

Sugarplum Mary's dialog:

> Hey, wouldja' mind helping me get into my point-of-sale terminal? It's down, and we kinda' need it running. Problem is: it is asking for a password. I never set one! Can you help me figure out what it is so I can get set up? Shinny says this might be an Electron application. I hear there's a way to extract an ASAR file from the binary, but I haven't looked into it yet.

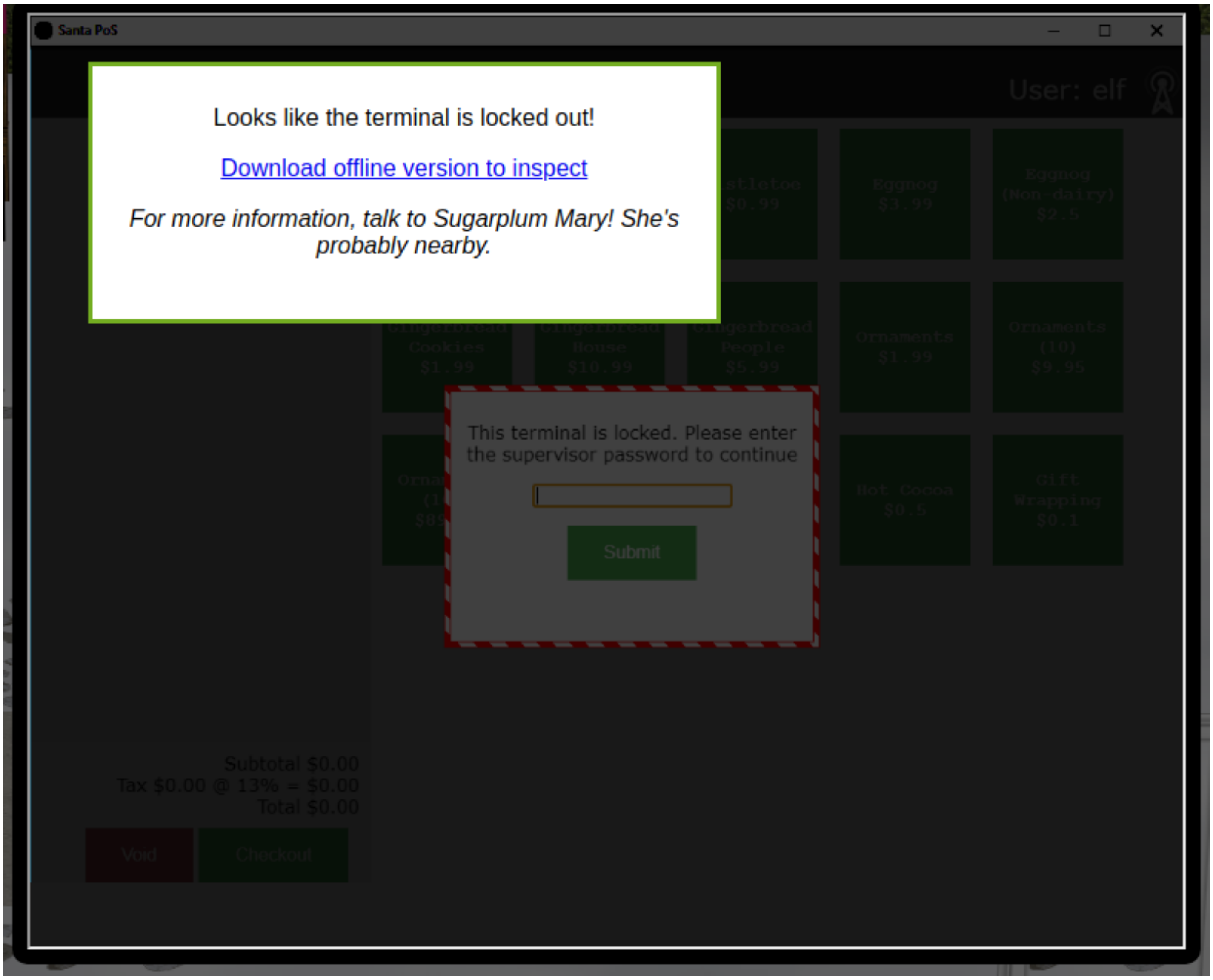
Hints

> It's possible to extract the source code from an Electron app. There are tools and guides explaining how to extract ASAR from Electron apps.

Solution

Electron is a framework for developing native applications with web technologies such as JavaScript, HTML, and CSS. From the guide on medium, it's possible to extract the source code of the application. We'll use the guide as a basis to finding and viewing the source code to the **Santa Shop** application.

Opening the **Santa Shop** terminal displays the following screen:



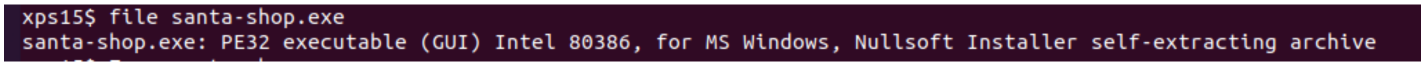
Looks like the terminal is locked out!

Download offline version to inspect

For more information, talk to Sugarplum Mary! She's probably nearby.

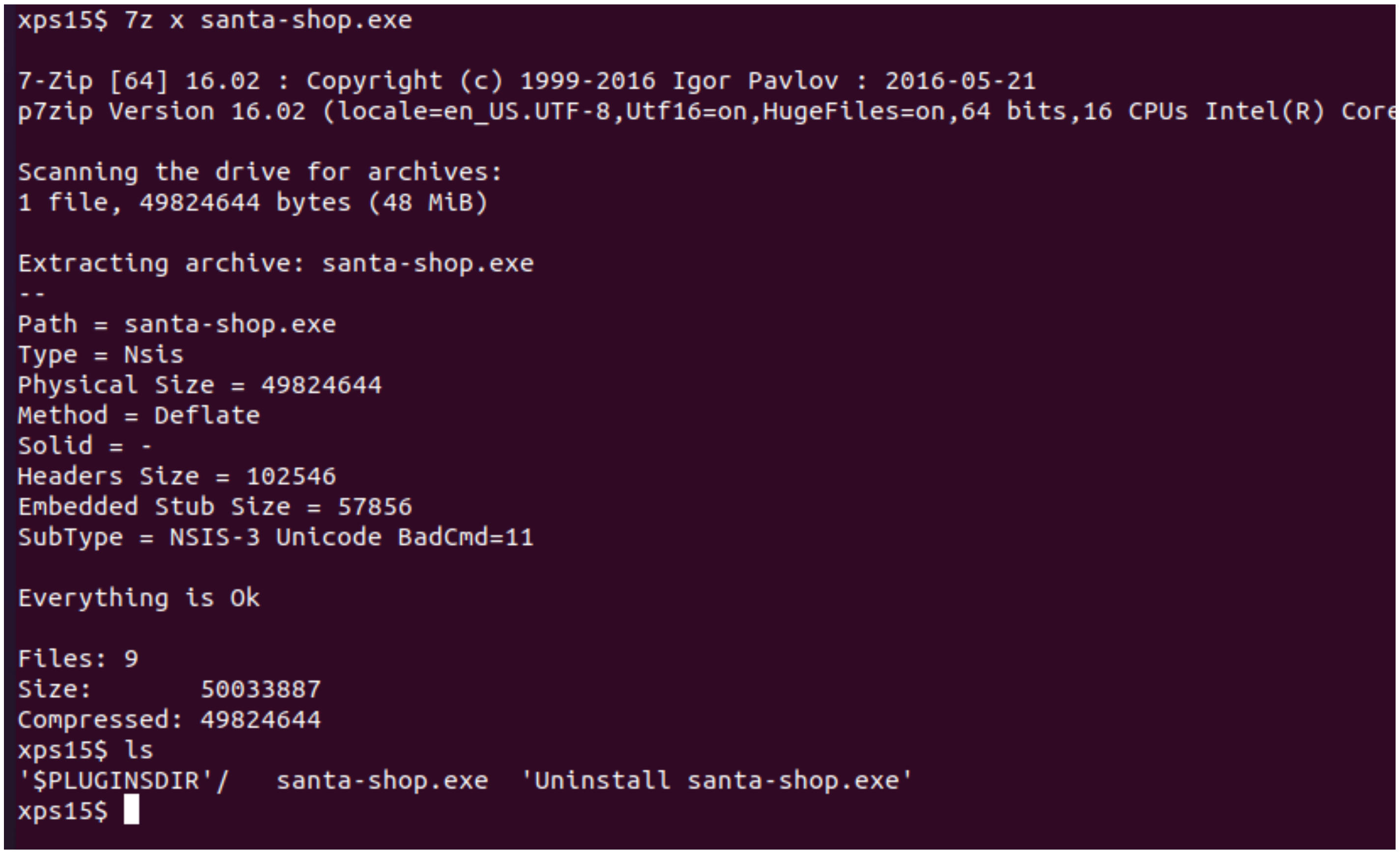
This terminal is locked. Please enter the supervisor password to continue

We're presented with a link to download the application for offline analysis. Downloading the file and running the `file` command on it gives us some details on what type of application we're dealing with:



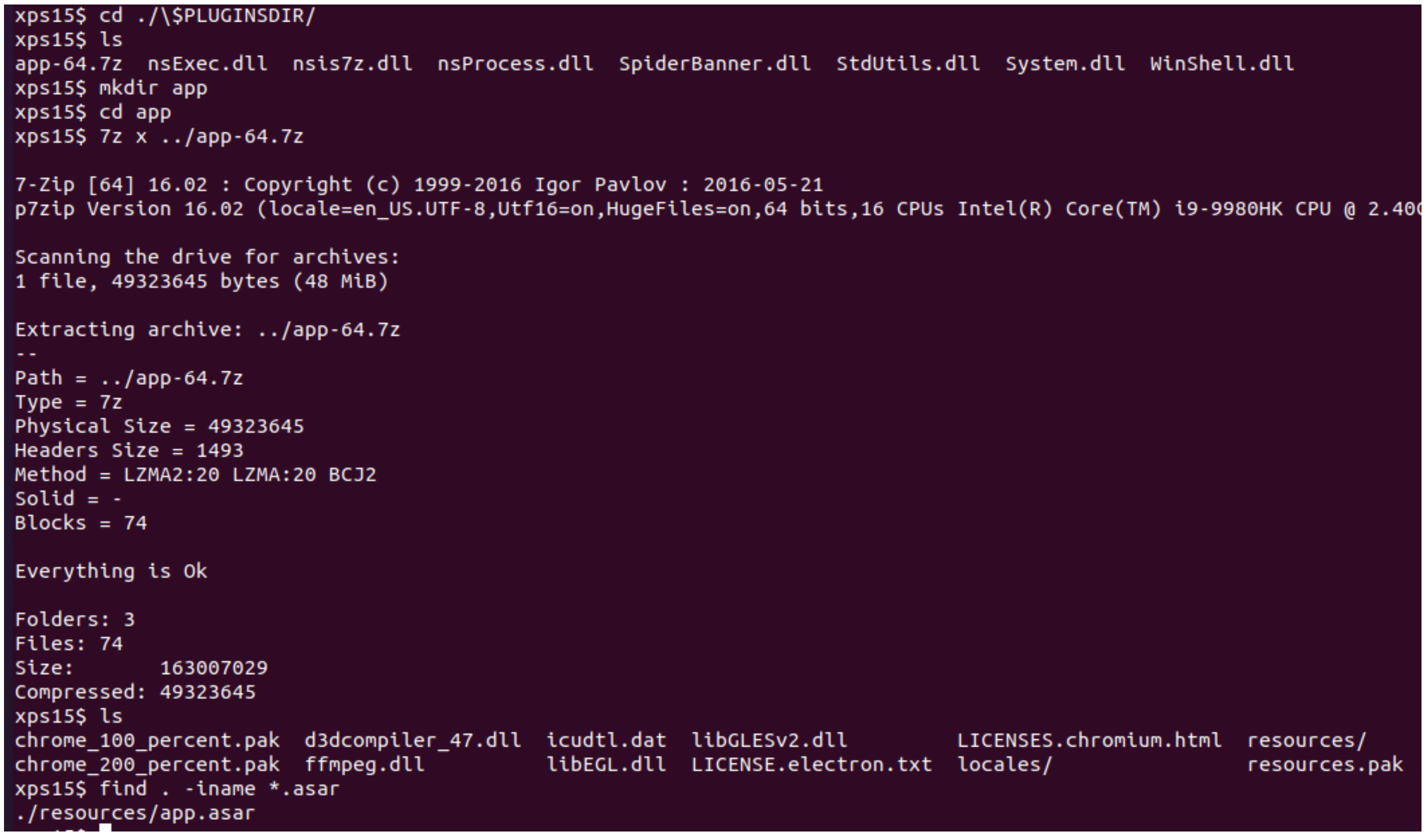
```
xps15$ file santa-shop.exe
santa-shop.exe: PE32 executable (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive
```

The important piece of information `file` returned is `Nullsoft Installer self-extracting archive`. While we could transfer the executable to a Windows machine and run the installer, it's easier to use a tool like `7zip` to just extract the installation files:



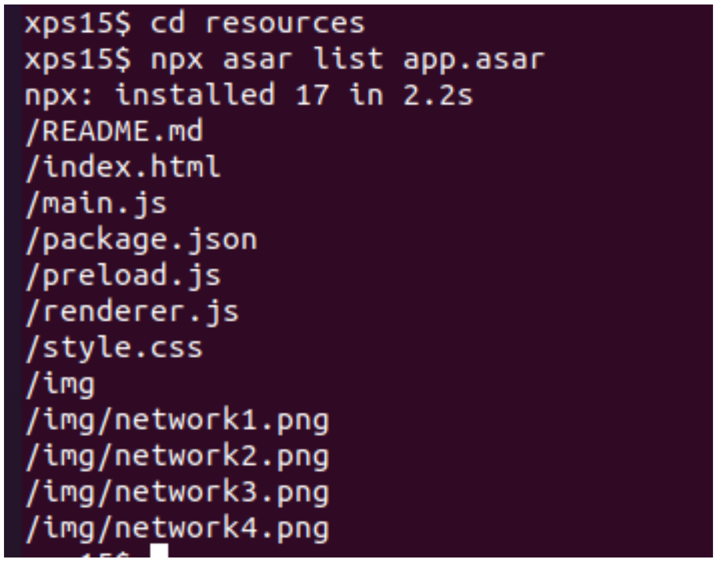
```
xps15$ 7z x santa-shop.exe

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,16 CPUs Intel(R) Core

Scanning the drive for archives:
1 file, 49824644 bytes (48 MiB)

Extracting archive: santa-shop.exe
--
Path = santa-shop.exe
Type = Nsis
Physical Size = 49824644
Method = Deflate
Solid = -
Headers Size = 102546
Embedded Stub Size = 57856
SubType = NSIS-3 Unicode BadCmd=11

Everything is Ok

Files: 9
Size:       50033887
Compressed: 49824644
xps15$ ls
'$PLUGINSDIR'/   santa-shop.exe  'Uninstall santa-shop.exe'
xps15$ 
```

This gives us the installer files, but unfortunately we don't yet have the `.asar` file that contains the application source. Looking in the `$PLUGINSDIR` directory, there is a `app-64.7z` file which looks promising. Let's create a directory to store it's contents, extract it with `7-zip`, and use the `find` command to look for any `.asar` files:



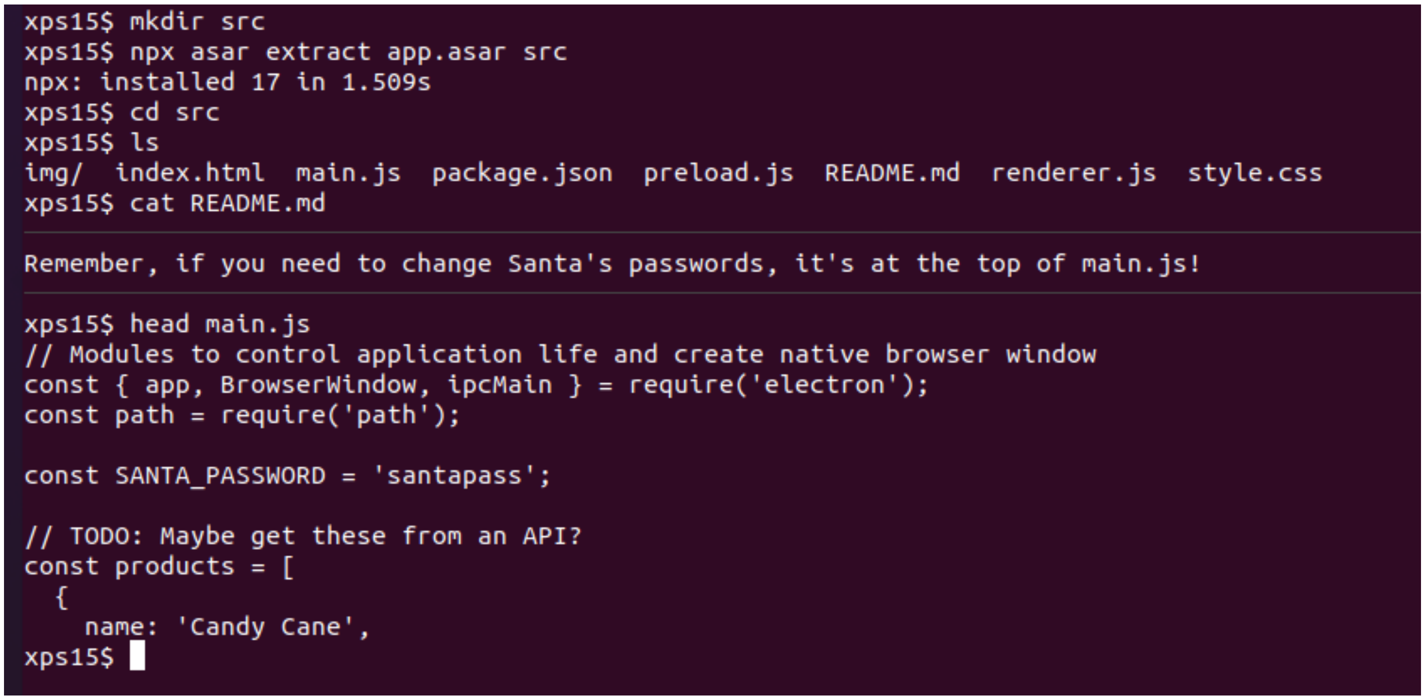
```
xps15$ cd ./\$PLUGINSDIR/
xps15$ ls
app-64.7z  nsExec.dll  nsis7z.dll  nsProcess.dll  SpiderBanner.dll  StdUtils.dll  System.dll  WinShell.dll
xps15$ mkdir app
xps15$ cd app
xps15$ 7z x ../app-64.7z

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,16 CPUs Intel(R) Core(TM) i9-9980HK CPU @ 2.40G

Scanning the drive for archives:
1 file, 49323645 bytes (48 MiB)

Extracting archive: ../app-64.7z
--
Path = ../app-64.7z
Type = 7z
Physical Size = 49323645
Headers Size = 1493
Method = LZMA2:20 LZMA:20 BCJ2
Solid = -
Blocks = 74

Everything is Ok

Folders: 3
Files: 74
Size:       163007029
Compressed: 49323645
xps15$ ls
chrome_100_percent.pak  d3dcompiler_47.dll  icudtl.dat  libGLESv2.dll         LICENSES.chromium.html  resources/
chrome_200_percent.pak  ffmpeg.dll          libEGL.dll  LICENSE.electron.txt  locales/                resources.pak
xps15$ find . -iname *.asar
./resources/app.asar
```

Aha, there is a file `app.asar` in the `resources` directory. From the guide, we need to use the `asar` utility from `node.js` to work with the file. After installing `node.js` and adding the `asar` command, we can run `npx asar list` command on `app.asar` to see a list of the application source code:



```
xps15$ cd resources
xps15$ npx asar list app.asar
npx: installed 17 in 2.2s
/README.md
/index.html
/main.js
/package.json
/preload.js
/renderer.js
/style.css
/img
/img/network1.png
/img/network2.png
/img/network3.png
/img/network4.png
```

`npx asar extract {filename} {directory}` is used to extract the source files from `{filename}` into `{directory}`. Extracting the source to a `src` directory and viewing the `README.md` tells us that the password is at the top of the file `main.js`:



```
xps15$ mkdir src
xps15$ npx asar extract app.asar src
npx: installed 17 in 1.509s
xps15$ cd src
xps15$ ls
img/  index.html  main.js  package.json  preload.js  README.md  renderer.js  style.css
xps15$ cat README.md

Remember, if you need to change Santa's passwords, it's at the top of main.js!

xps15$ head main.js
// Modules to control application life and create native browser window
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

const SANTA_PASSWORD = 'santapass';

// TODO: Maybe get these from an API?
const products = [
  {
    name: 'Candy Cane',
xps15$
```

And there is Santa's password, in cleartext in the application source code.

Answer

`santapass`