200312082

Introduction

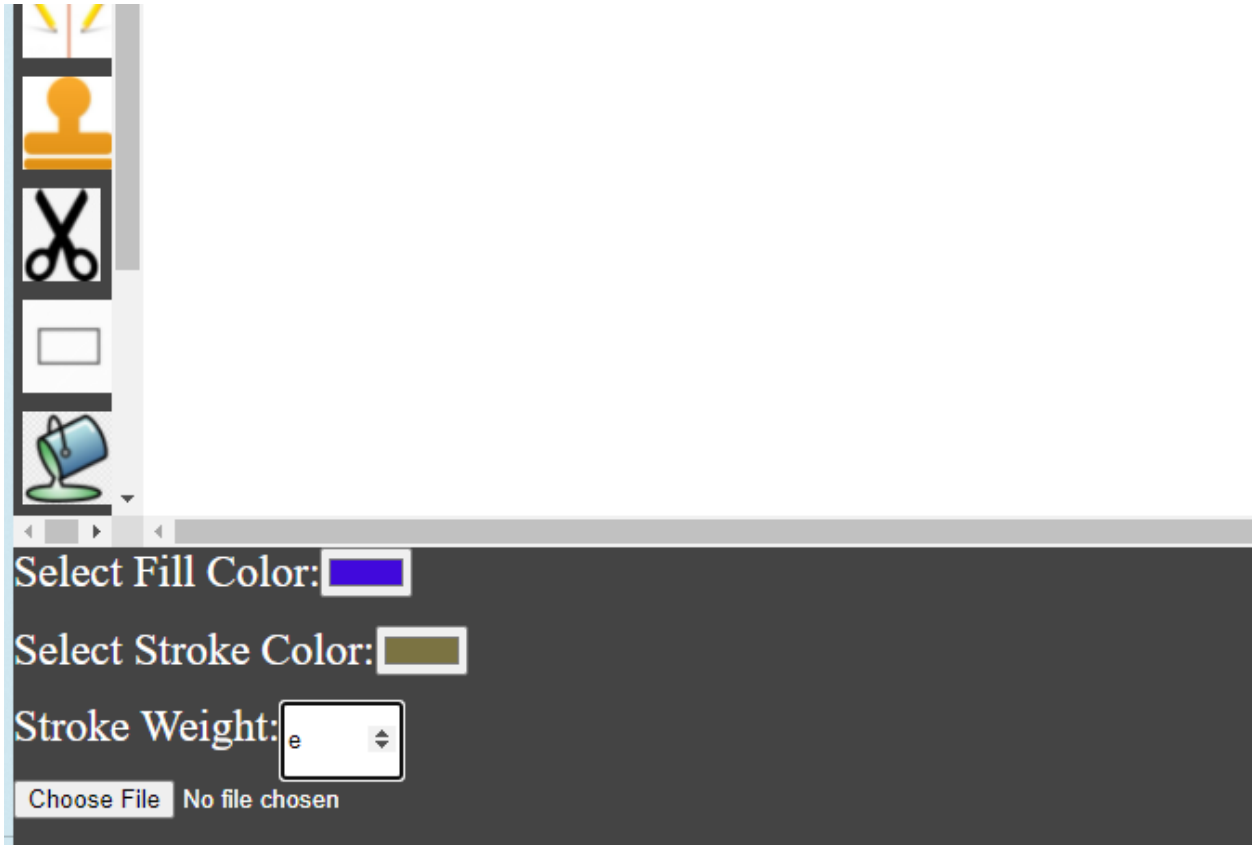
I have selected the drawing application from the “Introduction To Programming 2” module to analyse for secure programming.
The application was designed for drawing using tools organized into module, it was developed in javascript using p5js library

Recommendation 1 - Validate All Input

- Limit all numbers to the minimum (often zero) and maximum allowed values.

(Secure-Programs-HOWTO, page 44)

Problem



The program at some instances accepts input for it's normal operation. Examples of such inputs include; file for stamp tool and a number for determining the stroke weight of most of the tools like free hand, mirror draw, rectangle tool etc.

The issue with this input is that they are not thoroughly validated, though some validation has been done to some extent.

For instance;

- The stroke weight should only allow a number as it's input. Below is the code for the stroke weight input

```
49    <div class="box" id="sidebar"></div>
50    <div id="content" class="canvas-area"></div>
51    <div class="box component-area" id="initOpt">
52      <label for="color">Select Fill Color: </label>
53      <input type="color" name="color" id="color" value="#4109DC" />
54      <label for="color">Select Stroke Color: </label>
55      <input type="color" name="SC" id="SC" value="#7B7342" />
56      <label for="color"> Stroke Weight: </label>
57      <input type="number" name="SW" id="SW" value="1" min="1" max="50" />
58    </div>
59    <div class="box options">
60      <div id="options"></div>
```

Though the input specified the data type as "number" I noticed an anomaly that e/E is accepted as a number in the HTML input tag which is a serious security issue and it affects the performance of the application.

- File input:

5.6. File Names

> The names of files can, in certain circumstances, cause serious problems. This is especially a problem for secure programs that run on computers with local untrusted users, but this isn't limited to that circumstance. Remote users may be able to trick a program into creating undesirable filenames (programs should prevent this, but not all do), or remote users may have partially penetrated a system and try using this trick to penetrate the rest of the system.

(Secure-Programs-HOWTO, page 52)

Filenames that can especially cause problems include:

- Filenames with leading dashes (-). If passed to other programs, this may cause the other programs to misinterpret the name as option settings. Ideally, Unix-like systems shouldn't allow these filenames; they aren't needed and create many unnecessary security problems. Unfortunately, currently developers have to deal with them. Thus, whenever calling another program with a filename, insert "--" before the filename parameters (to stop option processing, if the program supports this common request) or modify the filename (e.g., insert "./" in front of the filename to keep the dash from being the lead character).
- Filenames with control characters. This especially includes newlines and carriage returns (which are often confused as argument separators inside shell scripts, or can split log entries into multiple entries) and the ESCAPE character (which can interfere with terminal emulators, causing them to perform undesired actions outside the user's control). Ideally, Unix-like systems shouldn't allow these filenames either; they aren't needed and create many unnecessary security problems.
- Filenames with spaces; these can sometimes confuse a shell into being multiple arguments, with the other arguments causing problems. Since other operating systems allow spaces in filenames (including Windows and MacOS), for interoperability's sake this will probably always be permitted. Please be careful in dealing with them, e.g., in the shell use double-quotes around all filename parameters whenever calling another program. You might want to forbid leading and trailing spaces at least; these aren't as visible as when they occur in other places, and can confuse human users.
- Invalid character encoding. For example, a program may believe that the filename is UTF-8 encoded, but it may have an invalidly long UTF-8 encoding. See Section 5.11.2 for more information. I'd like to see agreement on the character encoding used for filenames (e.g., UTF-8), and then have the operating system enforce the encoding (so that only legal encodings are allowed), but that hasn't happened at this time.
- Another other character special to internal data formats, such as "<", ";", quote characters, backslash, and so on.

(Secure-Programs-HOWTO, page 52)

```
10
11      //handle File picked
12      const handleFile = (file) => {
13        if (file.type === "image") {
14          createImg(file.data, "", "", (image) => {
15            img = image;
16            if (!slider) {
17              slider = createSlider(
18                5,
19                img.size().width * 1.5,
20                img.size().width / 2,
21                5
22              );
23              slider.parent(Gopt);
24              img.hide();
25            } else {
26              img.hide();
27            }
28          });
29        } else {
30          img = null;
31        }
32      };
```

The file input has been designed to allow images, but it is not detailed to the specific format of an image and acceptable name of file.

Solution

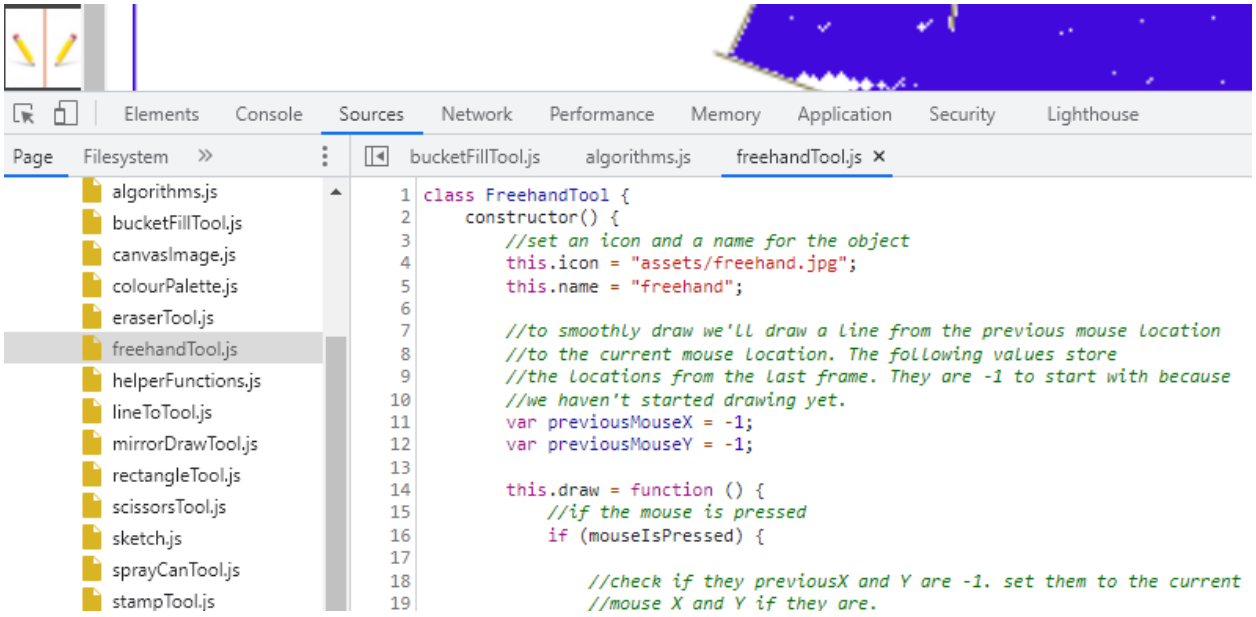
- The stroke weight input should be validated to ensure that only desired input is allowed. HTML input tag allows e/E because it stands for exponential which is not a desired input for this program. Therefore, a minimum and maximum value accepted should be added as an attribute to the input tag.
- Some image file format such as AVIF are not supported in all browser, all cross-platform format of an image accepted should be specified using a regular expression(*.jpg etc.) and the regex should only accept files with no leading dashes, Invalid character encoding, leading dashes, no spaces etc.

Recommendation 2 - Send Information Back Judiciously

Problem

> ### 9.2. Don't Include Comments
>
> When returning information, don't include any "comments" unless you're sure you want the receiving user to be able to view them. This is a particular problem for web applications that generate files (such as HTML). Often web application programmers wish to comment their work (which is fine), but instead of simply leaving the comment in their code, the comment is included as part of the generated file (usually HTML or XML) that is returned to the user. The trouble is that these comments sometimes provide insight into how the system works in a way that aids attackers.

(Secure-Programs-HOWTO, page 122)



Checking the source of this application while running in the developer tools exposes the comments of the internal workings of the program which is not intended for the user. This can help an attacker to easily hack the program.

Solution

Since the comments are not intended for the user and it doesn't help the usability of the program in any way, it should be totally removed when the program is about to go into production stage.

Recommendation 3 - Carefully Call Out to Other Resources

Problem

8.2. Limit Call-outs to Valid Values

Ensure that any call out to another program only permits valid and expected values for every parameter. This is more difficult than it sounds, because many library calls or commands call lower-level routines in potentially surprising ways. For example, many system calls are implemented indirectly by calling the shell, which means that passing characters which are shell metacharacters can have dangerous effects. So, let's discuss metacharacters.

(Secure-Programs-HOWTO, page 114)

```
14          createImg(file.data, "", "", (image) => {
15            img = image;
16            if (!slider) {
17              slider = createSlider(
18                5,
19                img.size().width * 1.5,
20                img.size().width / 2,
21                5
22              );
23              slider.parent(Gopt);
24              img.hide();
25            } else {
26              img.hide();
27            }
28          });
29        } else {
30          img = null;
31        }
32      };
```

The createImg function is part of the p5js library API and it's used for rendering images as p5.Elements on the canvas.

This function can make the program crash if the size of the data passed as it's first argument is bigger and it can not be converted to blob, this will also affect the program performance.

Solution

- The argument passed to createImg must be checked to ensure that the size is not too large.
- The file can be firstly converted to streams of byte using FileReader(Javscript Api) before using it as an argument.