Dokumentation Steganographytool

Auftraggeber	Daniel Brodbeck
Projektleiter	Yassin Abdelhamid & Samuel Leder
Autoren	Yassin Abdelhamid & Samuel Leder
Klassifizierung	Nicht klassifiziert
Status	Abgeschlossen, Genehmigt

Änderungsverzeichnis

Datum	Version	Änderung	Autoren
24.10.2021	1.0	Dokumentation erstellt	Yassin Abdelhamid & Samuel Leder

Inhaltsverzeichnis

Einleitung

Seit jeher gibt es Nachrichten welche die sehr vertraulicher Natur sind. Für solche Nachrichten gab es schon immer spezielle Wege und Möglichkeiten sie jemandem zukommen zu lassen. In der heutigen modernen Welt sind diese Möglichkeitem umso zahlloser. Eine davon ist das Verstecken einer digitalen Datei in einer anderen, auch Steganography genannt. In unserem Fall haben wir uns dafür entschieden einen Text in einem Bild zu verstecken. Unser Projekt beschäftigt sich damit eine Website zu erstellen die genau dies ermöglicht.

Projektaufbau und Eckdaten

Ausgangssituation / Problembeschreibung:

Unser Ziel ist es Daten in einem Bild zu verstecken.

Projektgesamtziel:

Eine Webapplikation, welches es dem User erlaubt ein Bild mit versteckten Daten zu füllen. Zusätzlich wird es dem User möglich sein das Endprodukt auszulesen. Per Upload Button kann ein User ein Bildformat JPG/JPEG oder PNG auswählen. In einem Dialogfeld kann eine versteckte Botschaft eingegeben werden, welche in die ausgewählte Bilddatei versteckt eingeschrieben wird sobald man den Kodier Button betätigt. Im Backend entsteht ein Bitmap, welche das Bild mit dieser verschlüsselten Nachricht zu einer neuen Bilddatei erstellt. Diese wird dem User wieder ausgegeben und man kann diese herunterladen.

Zusätzlich kann man eine Bilddatei welche schon eine verschlüsselte Nachricht enthält auch auslesen lassen. Der User geht wieder gleich vor, verwendet dafür jedoch den Dekodier Button und bekommt in einem Dialogfeld den ausgegeben Text zurück. Das ganze wird auf einer responsiven Webseite sein, welche einfach zu bedienen ist und überschaubar gestaltet sein wird.

Projektgruppe

Samuel Leder und Yassin Abdelhamid

Projektauftraggeber

Daniel Brodbeck

Projektdauer:

Geplanter Beginn: 13.09.2021

Geplantes Ende: 24.10.2021

Projektplanung:

- GitHub (Sourcecontrol System)
- WhatsApp (Koordinierung und Planung)
- Discord (Informationsautausch)

Projektorganisation:

Samuel Leder: Weboberfläche, Testing, Logik

Yassin Abdelhamid: Dokumentation, Weboberfläche, Anleitung

Projektressourcen:

Ressourcen: Beschreibung:

Personal: 2 Personen

Entwicklungsumgebung: Xampp, Visual Studio, Visual Studio Code

Programiersprachen: PHP / HTML / CSS / Javascript / C# (.NET Core) / Phyton

Geräte: PC

Projektrisiken und -unsicherheiten:

Schwierigkeitsgrad der Aufgabe aufgrund von fehlendem Vorwissen.

Funktionsweise & Konzept

Unterstützte Dateiformate:

Upload: JPG/JPEG, PNG

Download: PNG

Begründung:

Da einzelne Pixel ansprechbar und die Farbkanäle les- und editierbar sein müssen, sind nur die Formate .jpg/.jpeg und .png für den Upload valide. Als Download bereitgestellt wird das File als .png, da das dynamische Erstellen einer .png-Datei wesentlich einfacher ist als bei anderen Bilddateitypen.

Funktionsweise:

Beim Kodieren wird eine .jpg/.jpeg oder .png-Datei vom User hochgeladen. Anschliessend wird sie über ein «Canvas»-Objekt gelegt. Der Text wird dahinter auf einem weiteren Canvas dargestellt welcher die selben Abmessungen wie das Bild besitzt. Über ein Loop werden alle Pixel auf dem Textcanvas geprüft. Wenn ein schwarzer Pixel entdeckt wird, markiert der Algorithmus den Pixel auf dem Bild welcher die gleichen Koordinaten besitzt indem er den Wert auf dem grünen Farbkanal auf eine Sieben setzt. Das abgeänderte Bild wird dem User angezeigt und er kann es herunterladen.

Beim Dekodieren lädt der User eine .png-Datei hoch die bereits eine versteckte Nachricht erhält. Die Datei wird über einen Canvas mit den selben Abmessungen gelegt. Ein Loop prüft bei jedem Pixel ob der grüne Farbkanal den Zahlenwert Sieben besitzt. Ist dies so wird der Pixel auf dem Textcanvas an der selben stelle schwarz eingefärbt. Anschliessend wird dem User der Textcanvas angezeigt und die dekodierte Nachricht ist für ihn nun lesbar.

Prozeduren zur Aufbereitung, Optimierung und Integration von Dateien

Responsive Webdesign und Browserunterstützung

Der Bilderupload wird auf die Formate JPG/JPEG und PNG beschränkt. Diese Formate werden von allen Browsern unterstützt. Zusätzlich ist es durch responsive Webdesign möglich, dass die Weboberfläche sich je nach Gerätegrösse anpasst und weiterhin Benutzerfreundlich bleibt.

Im Folgenden Bild ist die Funktion der Upload Validierung zu sehen.
Diese ermöglicht es dem User keine anderen Dateien als JPG/JPEG und PNG hochzuladen.

```
function fileValidationEncode() {

    let fileInput = document.getElementById('encode');
    let filePath = fileInput.value;

    let allowedTypes = /(\.jpg|\.jpeg|\.png)$/i;

    if (!allowedTypes.exec(filePath)) {
        alert('Bitte wählen Sie eine dieser Dateien aus: jpeg, jpg, oder png!
');
        fileInput.value = '';
        return false;

    }
}
```

Bearbeitung der Bilder

Dieses Werkzeug erstellt ein Canvas mit Ihrem Bild und ein Canvas mit identischer Grösse mit Ihrem Text. Es durchsucht dann jeder Pixel des Canvas. Wenn es schwarz sieht, weiss es, dass das angezeigte Pixel Teil der Nachricht ist. Es findet das Pixel an derselben Stelle in Ihrem Canvas und stellt sicher, dass der Grünwert des RGB mit sieben endet.
Wenn es weiss oder transparent sieht, weiss es, dass es nicht im Text ist, und stellt sicher, dass identisch platzierte Pixel auf dem Grünwert Ihres Canas nicht mit sieben endet.

Nachdem dies für das gesamte Bild durchgeführt wurde, haben wir jetzt ein Bild, bei dem der Grünwert jedes Pixels nicht mit sieben endet, es sei denn, es wird eine Nachricht buchstabiert.

Die Dekodierungsfunktion kehrt dies um, sie durchsucht die Pixel des hochgeladenen Bildes und blendet alle Pixel aus, bei denen kein grüner Wert mit der Endung sieben gefunden wird.

Hier die Funktion zum kodieren:

```
function handleImageEncode(e){

    let readerEncode = new FileReader();
    readerEncode.onload = function(event){
        let imgToEncode = new Image();
        imgToEncode.onload = function(){
            encodeCanvas.width = imgToEncode.width;
            encodeCanvas.height = imgToEncode.height;
            textCanvas.width=imgToEncode.width;
            textCanvas.height=imgToEncode.height;
            tctx.font = "30px Arial";
```

```
      let messageText = (messageInput.value.length) ? messageInput.value :
'Error: Bitte erneut versuchen und zuerst die Geheimnachricht eingeben';
            tctx.fillText(messageText,10,50);
            ctx.drawImage(imgToEncode,0,0);
            let imgData = ctx.getImageData(0, 0, encodeCanvas.width,
encodeCanvas.height);
            let textData = tctx.getImageData(0, 0, encodeCanvas.width,
encodeCanvas.height);
            let pixelsInMsg = 0;
                pixelsOutMsg = 0;
            for (let i = 0; i < textData.data.length; i += 4) {
                if (textData.data[i+3] !== 0) {
                    if (imgData.data[i+1]%10 == 7) {
                    }
                    else if (imgData.data[i+1] > 247) {
                        imgData.data[i+1] = 247;
                    }
                    else {
                        while (imgData.data[i+1] % 10 != 7) {
                            imgData.data[i+1]++;
                        }
                    }
                    pixelsInMsg++;
                }
                else {
                    if (imgData.data[i+1]%10 == 7) {
                        imgData.data[i+1]--;
                    }
                    pixelsOutMsg++;
                }
            }
            console.log('pixels within message borders: '+pixelsInMsg);
            console.log('pixels outside of message borders: '+pixelsOutMsg);

            ctx.putImageData(imgData, 0, 0);
        };
        imgToEncode.src = event.target.result;
    };
    readerEncode.readAsDataURL(e.target.files[0]);
}
```

Die Funktion für Dekodieren:

```
function handleImageDecode(e){

    console.log('Decoding Image');
    let readerToDecode = new FileReader();
    readerToDecode.onload = function(event){
        console.log('readerToDecode loaded');
```

```
        let imgToDecode = new Image();
        imgToDecode.onload = function(){
            console.log('imgToDecode loaded');
            decodeCanvas.width = imgToDecode.width;
            decodeCanvas.height = imgToDecode.height;
            dctx.drawImage(imgToDecode,0,0);
            let decodeData = dctx.getImageData(0, 0, decodeCanvas.width,
decodeCanvas.height);
            for (let i = 0; i < decodeData.data.length; i += 4) {
                if (decodeData.data[i+1] % 10 == 7) {
                    decodeData.data[i] = 0;
                    decodeData.data[i+1] = 0;
                    decodeData.data[i+2] = 0;
                    decodeData.data[i+3] = 255;
                }
                else {
                    decodeData.data[i+3] = 0;
                }
            }
            dctx.putImageData(decodeData, 0, 0);
        };
        imgToDecode.src = event.target.result;
    };
    readerToDecode.readAsDataURL(e.target.files[0]);
}
```

Usability

Durch angeschriebene Knöpfe und Infotexten, sowie auch «Altermessages», ist die Anwendung dieser Webseite sehr benutzerfreundlich und Kinderleicht.

Im Codesnippet ist ein Infotext zur Verwenung der Webseite zusehen:

```
    <p style="border: 1px solid; background-color: rgba(231, 230, 230, 0.11);
padding: 5px;">
        Auf dieser Webseite ist es Ihnen möglich, eine Geheimnachricht in ein
Bild mit Format JPEG/JPG oder PNG zu
        verstecken.<br>
        Folgenderweise müsse Sie vorgehen um das Steganography Tool zu nutzen:
<br>
        1. Geben Sie Ihre Geheimnachricht im Textfeld ein. <br>
        2. Sende Sie Ihre Geheimnachricht durch den "ok" Knopf ab. <br>
        3. Es erscheint ein neuer Knopf um eine gewünschte Bilddatei
hochzuladen. Ihre Geheimnachricht wird in der
        zuhochladenden Bilddatei versteckt. <br>
```

```
        4. Die neue kodierte Datei wird angezeigt. Laden Sie diese herunter.
<br>
    </p>
```