인터넷 게시물 채증 프로그램 제작

Track	Digital Forensic
Category	Tech_04
Nick Name	L3ad0xFF

목 차

1. 개발 목적 및 목표 1

2. 개발 SCRIPT – CHROME EXTENSION 1

2.1 MANIFEST.JSON 1

2.1.1 기본 정보 입력 2

2.1.2 browser_action 2

2.1.3 Permissions 2

2.2 POPUP.HTML & STYLE.CSS 3

2.2.1 Html.header 3

2.2.2 Html.body 3

2.2.3 style.css 6

2.3 MAIN.JS 7

2.3.1 EventListener 7

2.3.2 Button target c_screentshot 함수 7

2.3.3 Current_download 함수 8

2.3.4 cur_handleFileSelect 함수 9

2.3.5 getLocalTime4Exif 함수 9

2.3.6 Button id_0 f_screentshot 함수 10

2.3.7 captureToBlobs 함수 11

2.3.8 initiateCapture 함수 11

2.3.9 getBlobs 함수 11

2.3.10 capture 함수 12

2.3.11 _initScreenshots 함수 13

2.3.12 __filterScreenshots 함수 13

2.3.13 DB 관련 함수 14

2.4 PAGE.JS 15

2.5 PIEXIF.JS 16

2.6 NOTICE_DB_PATHHTML & STYLE2.CSS 16

3. 확장 프로그램 18

4. 참고문헌 21

1. 개발 목적 및 목표

- 현재 브라우저의 화면을 활성화 된 상태 그대로 저장하기 위함이다.
- 브라우저 확장 또는 독립 프로그램으로 개발이 가능하다
- 현재 활성화 된 브라우저를 캡처할 경우, 현재 화면 상태로 저장하거나 스크롤하여 전체 페이지를 저장 할 수 있도록 하며, 옵션으로 조정한다.
- 캡처 실행 시, 활성화 된 화면의 소스파일, 채증화면을 저장한다.
- 채증화면 저장 시, 별도의 표준시 정보를 채증화면 저장파일의 EXIF에 시간정보를 같이 저장한다.
- 별도의 DB에 시간, 파일명, MD5 등을 저장한다.

2. 개발 Script – Chrome Extension

2.1 Manifest.json

- Manifest.json 파일은 확장 프로그램에 대한 중요한 정보(이름 및 필요한 권한 등)를 Chrome에게 알리는 역할을 한다.
- 가능한 한 manifest.json을 이용하여 확장 할 수 있는 범위는 manifest.json 파일이 있는 디렉토리를 기준으로 하위 폴더 및 함께 있는 파일들이다.
- 2014년 1월부터 manifest.json의 version 1은 지원되지 않기 때문에 version 2를 사용하도록 한다.

```
{
  "name": "BoB - Screenshot, Use for Evidence",
  "version": "1.3",
  "description": "Gather evidence",

  "browser_action": {
    "default_icon": "image/title.png",
    "default_title": "Screen Collector",
    "default_popup" : "popup.html"
  },
  "permissions": [
    "activeTab",
    "webRequest",
    "contentSettings",
    "desktopCapture",
    "downloads",
    "pageCapture",
    "<all_urls>",
    "unlimitedStorage"
  ],
  "manifest_version": 2
}
```

2.1.1 기본 정보 입력

- name, version, description 3 항목은 확장 프로그램을 식별할 수 있는 항목들이며, 사용자가 임의로 입력을 수행한다.

Manifest.json에 입력한 항목들에 대한 값이 chrome extension 등록 시, 출력된다.

2.1.2 browser_action

- 확장 프로그램을 등록하면 주소 표시 줄 옆 작은 아이콘이 생성된다. 해당 아이콘을 사용자가 누를 시, 어떤 동작을 할 것인가에 대한 옵션을 설정한다.
  - Default_icon : image/title.png
  - Default_title : "Screen Collector"
  - Default_popup : "popup.html"

  확장 프로그램을 등록 시, 주소 표시줄 옆에 생성되는 아이콘의 그림을 manifest.json이 존재하는 디렉토리 내의 image 폴더 내부의 title.png로 설정한다..

  주소 표시줄 옆에 생성되는 아이콘에 마우스 커서를 올리면 출력되는 확장 프로그램의 제목을 "Screen Collector"로 설정한다.

  확장 프로그램을 실행 시, 창이 생성되도록 할 경우 설정하는 추가 항목이며, mesifest.json과 같은 경로에 존재하는 popup.html을 확장 프로그램 실행 시 나타나도록 설정한다.

2.1.3 Permissions

- 해당 확장 프로그램을 사용 할 때, 허용 되는 조건들을 설정하는 영역이다.
- Html5와 구글 확장 프로그램 정책에 의해 javascript에서 구글 api가 실행 되기 위해서는 javascript에 작성하는 브라우저 관련 api 함수들의 실행이 가능하도록 허용한다.
- 캡처 하고자 하는 화면의 정보에 관한 허용
  - activeTab, webRequest, contentSettings, desktopCapture, downloads, pageCapture
- 소스 코드 확보 및 채증 대상 URL 접근 허용을 위한 허가 정책
  - <all_urls>
- Web SQL 데이터베이스 접근 허용 정책
  - unlimitedStorage

2.2 popup.html & style.css

- manifest.json을 이용하여 확장 프로그램을 등록하고, 확장 프로그램을 실행 하기 위해 사용자 클릭이 이루어지면 생성되는 popup 창에 대한 설정
- 2.1항 manifest.json에서 설정한 default_popup 항목에 설정한 html

2.2.1 Html.header

- 해당 html의 design을 설정하기 위해 연동할 style.css를 선언한다.

```
<!DOCTYPE html>
<html><head lang="en">
  <meta charset="UTF-8">
  <title>BoB Digital Forensic 6th 3rd_Stage</title>
  <link href="http://fonts.googleapis.com/
  css?family=Open+Sans:400,700" rel="stylesheet" type="text/css">
  <link href="style.css" rel="stylesheet">
</head>
```

2.2.2 Html.body

- Html을 이용하여 동작 시킬 스크립트를 호출하기 위해, javascript 파일을 기재한다.
- Html5부터 보안정책으로 html에 직접 script 삽입이 기본적으로 불가능하다.

```
<body class="popup-box">
<script src="./main.js"></script>
<script src="./piexif.js"></script>
<div class="window_header">
  <h2>Browser Screenshot</h2>
  <p>Best of The Best 6th Digital Forensic</p>
</div>
```

popup창을 하나의 ''popup-box' 이름의 class로 선언한다. 위의 위치부터 나누어서 'Browser Screenshot' 제목과 부가 설명의 string을 입력한다.

현재 html과 같은 경로의 main.js와 piexif.js를 html이 실행될 때 동시에 호출한다.

- ◆ main.js : 현재 화면 캡처 및 전체 화면 캡처를 위한 스크롤 및 캡처
- ◆ piexif.js : 화면 캡처를 저장하기 전에 EXIF 정보를 생성 또는 수정

➢ html에서 DB에 접근하기 위해 사용자 컨트롤

```
    <a class="About_DB">
      <p> DB Name : <input type="text" id="dbname" value=" "></p>
      <button id_1="DB_Create_Load">Create or Load DB</button>
    </a>
</li>
<li>
    <a class="About_DB">
      <p> Table Name : <input type="text" id="t_name" value=" ">
      </p>
      <button id_3="Table_Create_Load">Create or Load Table</
      button>
    </a>
</li>
<li>
  <a class ="About_DB">
    <p> Del Table Name : <input type="text" id="td_name" value="
    "></p>
   <button id_4="Table_Del">Delete Table</button>
 </a>
</li>
<li>
  <a class ="About_DB">
   <span> Please set DB, Table before show</span><button id_5="
   showDB">Show DB Data</button>
 </a>
</li>
<li>
  <a class = "About_DB">
    <p> <textarea id="print_window" cols="40" rows="8" readonly="
    readonly"></textarea></p>
  </a>
</li>
<li>
```

"About_DB"라는 하나의 Class로 각 DB에 수행할 기능들을 묶는다.

DB 생성, Table 생성, Table 삭제의 경우 각각의 입력창을 제공하고, 사용자 입력의 값에 맞는 query문을 전송하도록 설정한다. 전송을 위한 사용자 옵션은 html에서 기본 제공하는 button 형식으로 click을 할 수 있도록 한다.

DB 내용 호출의 경우, 사용자가 설정한 DB와 Table의 있는 정보를 text 창에 출력 할 수 있도록 설정한다. 호출 여부 또한 button을 이용하여 사용자에게 접근성을 제공한다.

각각의 button은 모두 다른 button이 되도록 식별 선언을 각각 다르게 설정한다. Button의 식별 값이 같을 경우 하나의 button을 click하면 같은 식별 값을 같는 button 또한 같은 기능을 수행한다.

DB 생성 및 호출 : button id_1 / Table 생성 및 호출 : button id_3 / Table 삭제 : button id_4 / DB 내용 호출 : button id_5

➢ html에서 현재 화면 또는 전체 화면 캡처 옵션 제공

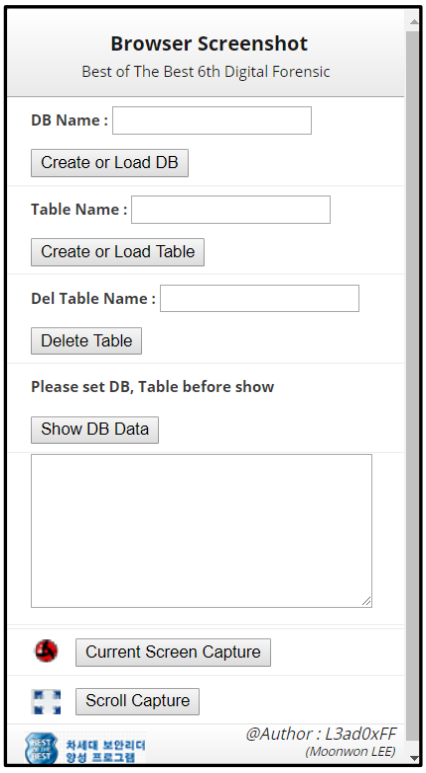
```
<li>
    <a class="capture-visible">
      <span class="icon icon1"></span> <a> <button target="
      current_shot">Current Screen Capture</button>
    </a> </a>
  </li>
  <li>
    <a class="capture-all">
      <span class="icon icon2"></span> <a> <button id_0="
      whole_page">Scroll Capture</button>
    </a> </a>
  </li>
</ul>
</div>
```

DB와 관련된 button의 경우, text 공간을 제공했다면, Capture 옵션을 제공할 때는 각각의 button에 icon image를 삽입한다.

Icon class를 선언하여 icon1과 icon2를 모두 관리한다. 모두 button 앞에 icon을 제공하고, 각각 icon에 제공할 삽입 그림은 icon1, icon2를 이용하여 관리한다.

현재 활성화 된 브라우저에서 사용자의 화면만 캡쳐와 전체 크기에 맞추어 캡쳐를 두 개의 button으로 옵션을 제공하고 각각의 button의 식별 값을 다르게 하여, 사용자 click시 원하는 결과에 맞는 결과가 실행되도록 분리한다.

현재 화면 캡쳐 : button target / 전체 화면 캡처 : botton id_0



Html Body에서 첫 div 부분
제목과 부가 설명 선언

- DB에 관련하여 사용자 컨트롤이 가능한 부분
- DB 생성 및 호출, Table 생성 및 호출, Table 삭제 요청 시, 대상 DB 또는 Table을 입력할 수 있는 입력창을 제공.
- DB Data 출력 button을 click하면 해당 button 아래의 text창에 DB Table의 데이터가 출력되도록 한다.

현재 화면 캡처와 전체 화면 캡처를 사용자가 선택할 수 있도록 2가지 button 옵션을 제공

2.2.3 style.css

➢ html의 Design을 결정하는 요소

```
.popup-box p {
    padding: 0;
    margin: 0;
}

.window_header {
    padding: 2px 10px 12px 10px;
    box-shadow: inset 0 1px 0 rgba(255,255,255,0.2);
    border-bottom: 1px #aaaaab solid;
    border-radius: 8px 8px 0 0;
    background: #ffffff;
    background: -moz-linear-gradient(top, #ffffff 0%, #eaeaea 100%);
    background: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #ffffff), color-stop(100%, #eaeaea));
    background: -webkit-linear-gradient(top, #ffffff 0%, #eaeaea 100%);
    background: -o-linear-gradient(top, #ffffff 0%, #eaeaea 100%);
    background: -ms-linear-gradient(top, #ffffff 0%, #eaeaea 100%);
    background: linear-gradient(to bottom, #ffffff 0%, #eaeaea 100%);
 filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffffff', endColorstr='#eaeaea', GradientType=0 );
    font-size: 13px;
}

.window_header h2 {
    font-size: 16px;
    font-weight: 700;
    color: #191919;
    margin: 0;
    padding: 13px 0 0 73px;
}
.window_header p {
    font-size: 12px;
    font-weight: 500;
    color: #191919;
    margin: 0;
    padding: 3px 0 0 50px;
}
```

Popup의 전체 크기 설정과 window_header라 선언한 div class에 관한 design 설정 값 회색 바탕의 색상과 위, 아래 사이 간격, 제목과 부가설명의 글자 크기 등을 설정한다.

```
.window_nav ul li a {
    display: block;
    /*color: #409922;*/
    /*background: #2a84ff;*/
    line-height: 34px;
    text-decoration: none;
    border-bottom: 1px #eee solid;
    padding: 1px 19px;
    font-size: 12px;
    font-weight: 700;
}
.window_nav ul li .icon1 {
    background: url(image/icon1.png) no-repeat center;
}
.window_nav ul li .icon2 {
    background: url(image/icon2.png) no-repeat center;
}
```

각 button이 제공되는 영역의 text 입력 창 또는 icon 제공 관련 요소이며, 삽입될 icon의 경로가 해당 요소에 지정되어 입력된 경로로부터 icon 삽입 그림을 호출하여 html에서 출력한다.

2.3 Main.js

➢ 실질적으로 메인으로 동작하는 javascript이며, html에서 제공되는 button을 사용자 click시, 각 구별된 button의 행위에 맞는 결과가 도출되도록 실행한다.

2.3.1 EventListener

```
document.addEventListener('DOMContentLoaded', function () {
  document.querySelector('button[target]').addEventListener('click', c_screentshot);
  document.querySelector('button[id_0]').addEventListener('click', f_screentshot);
  document.querySelector('button[id_1]').addEventListener('click', createORloadDB);
  // document.querySelector('button[id_2]').addEventListener('click', loadDB);
  document.querySelector('button[id_3]').addEventListener('click', createOrloadTable);
  document.querySelector('button[id_4]').addEventListener('click', DeleteTable);
  document.querySelector('button[id_5]').addEventListener('click', showDB);
});
```

Html5부터 html에서 button onclick을 이용하여 바로 button에 연결된 script 실행이 정책적으로 불가능.

javascript에서 button click의 event 발생 여부를 기다리고 button이 사용자에 의해 click 되는 signal이 발생하면 눌린 button을 button 식별 값에 의해 구별하고 연결된 함수를 실행한다.

Button click에 따른 함수 실행은 아래와 같다.
button target : c_screenshot / button id_0 : f_screenshot / button id_1 : creatORloadDB
button id_3 : creatOrloadTable / button id_4 : DeleteTable / button id_5 :showDB

2.3.2 Button target c_screentshot 함수

➢ 현재 사용자에게 제공되는 화면 크기만큼 캡처를 제공한다.

```
function c_screentshot(element) {
    if(!!window.openDatabase) {
     alert("현재 브라우저는 Web SQL Database를 지원합니다")
    }
    else{
      alert("현재 브라우저는 Web SQL Database를 지원하지 않습니다")
    }
   // var downloadLink = document.querySelector("#MHTML");
    chrome.tabs.getSelected(null, function(tab) {
        chrome.pageCapture.saveAsMHTML({tabId: tab.id}, function (mhtml){
            var url = window.URL.createObjectURL(mhtml);
            src_name = getLocalTimeString()+" "+tab.url.split("://")[1] + ".mht"
            current_download(url, src_name)
        });
    });

    chrome.tabs.getSelected(null, function(tab) {
        chrome.tabs.captureVisibleTab(function(screenshotUrl) {
            //alert (screenshotUrl)
            target_url = tab.url
            cur_exif = cur_handleFileSelect(screenshotUrl)
            downname = getLocalTimeString() + " currnet_("+ target_url + ").jpg"
            db_hash_filename = getLocalTimeString()+" "+calcMD5(atob(cur_exif.split(',')[1])).toUpperCase()+".jpg"
            current_download(cur_exif, downname);
            InsertData_capture(downname, db_hash_filename);
        });
    });
}
```

- 현재 활성화 된 브라우저에서 Web SQL 지원 여부를 확인한다.
- 캡쳐를 진행할 경우 캡쳐 할 창이 활성화 되어 있을 것이며, chrome.tabs api를 이용하여 활성화 된 창 (구글의 경우 하나의 창에서 여러 개의 url을 tab을 이용하여 구분)의 tab 정보를 얻어온다.
- 현재 화면 캡처를 진행 할 때 소스파일 저장도 추가로 진행 되어야 한다.
- 현재 활성화 된 tab의 url에 해당되는 데이터를 mhtml 형식의 포멧으로 저장하기 위해 chrome.tabs api의 함수를 호출한다. url을 그대로 chrome에서 mthml 포멧 형태로 변환 저장하고 변환된 데이터를 사용자의 PC로 저장할 수 있도록 javascript 내의 download 함수를 호출한다. Download 함수는 데이터와 저장 시, 파일명을 인자로 하며, mhtml의 파일명은 해당 포멧을 지원하는 mht 확장자로 지정한다.
- mht 확장자 파일은 캡쳐 당시 소스코드를 동적 js까지 수집하고, 하나의 브라우저 창으로 변환하여 저장한다. 실행 시 IE로 캡쳐 당시의 url이 그대로 실행되며, string을 확인하면 html 소스코드가 존재함을 확인할 수 있다.
- 현재 화면 캡처는 chrome.tabs api 중 현재 활성화 된 화면에 맞게 그림파일로 변환하는 함수를 호출한다.(captureVisibleTab)
- captureVisiblaTab 함수는 입력된 데이터를 jpg 파일 포멧으로 변환한다.
- 브라우저 창에 출력된 화면이 모두 bas64의 string stream date로 변환되고 해당 string stream은 cur_handleFileSelect 함수의 인자로 사용되고 해당 함수의 return 값이 cur_exif에 저장된다.
- cur_exif에 저장된 데이터는 내부 download 함수에 의해 jpg 형식으로 저장이 된다.

2.3.3 Current_download 함수

➢ 현재 화면 캡처 후 사용자의 PC로 저장하기 위해 호출되는 함수이다.

```
function current_download(capture_url, filename) {
  var a = document.createElement("a");
  a.href = capture_url;
  a.setAttribute("download", filename);
  var b = document.createEvent("MouseEvents");
  b.initEvent("click", false, true);
  a.dispatchEvent(b);
  return false;
}
```

- a라는 요소를 생성하고, href 속성으로 데이터를 링크 속성으로 생성한다.
- 마우스 이벤트로 a href를 누르면 download가 실행되고, 파일명으로 링크 속성으로 설정한 데이터가 저장되는 알고리즘으로 다운로드를 구성한다.

2.3.4 cur_handleFileSelect 함수

➢ base64로 인코딩된 string stream data에 EXIF 속성을 추가하도록 하는 함수

```
function cur_handleFileSelect(dataURI) {
    var f = dataURI;
    // make exif data
    var zerothIfd = {};
    var exifIfd = {};
    var gpsIfd = {};
    //alert (getLocalTime4Exif());
    zerothIfd[piexif.ImageIFD.Software] = "L3ad0xFF";
    exifIfd[piexif.ExifIFD.DateTimeOriginal] = getLocalTime4Exif();
    var exifObj = {"0th":zerothIfd, "Exif":exifIfd, "GPS":gpsIfd};
    // get exif binary as "string" type
    var exifBytes = piexif.dump(exifObj);

    var reader = new FileReader();

    var origin = piexif.load(dataURI);

    // insert exif binary into JPEG binary(DataURL)
    var jpeg = piexif.insert(exifBytes, dataURI)
    return jpeg;
}
```

- piexif에서 각 요소들을 호출한다. piexif는 html에서 main.js와 함께 선언한 piexif.js를 의미한다.
- EXIF의 각 태그에 값을 설정한다. 과제에서 원하는 별도의 표준 시 시간은 getLocatTIme 4Exif 함수에서 얻어온다.
- EXIF를 base64로 인코딩 된 캡처 대상의 string stream data에 밀어 넣고, 결과 값을 함수의 return값으로 반환한다.
- 반환된 값이 current_download에 의해 사용자 PC에 jpg 형식으로 저장되며, 이 함수에서 설정한 EXIF 값이 삽입되어 있다.

2.3.5 getLocalTime4Exif 함수

➢ EXIF의 시간 Tag에 넣을 값을 해당 함수의 반환 값으로 사용하였다.

```
function getLocalTime4Exif() {
    var now_time = new Date();
    var year = now_time.getFullYear();
    var month = now_time.getMonth() +1 ;
    var date = now_time.getDate();
    var hour = now_time.getHours();
    var minute = now_time.getMinutes();
    var second = now_time.getSeconds();
    var fulltime = year + ":" + month + ":" + date + " " + hour + ":" + minute + ":" + second;

    return fulltime;
}
```

- 현재 시간 값을 new Date로 가져오는 내장 함수를 사용하였다. 내장함수의 반환 값은 해당 코드를 호출하는 사용자의 로컬 컴퓨터에 설정된 표준 시간대에 해당한다.
- 호출한 시간 값에서 각각 원하는 부분만 따로 분류하고, EXIF Tag 값에 알맞은 형식에 맞도록 문자열로 변환한다.
- Month의 경우 0의 값이 1월로 설정되어 있기 때문에 원하는 month value를 도출하기 위해서는 +1을 하여 변수를 저장해야 한다.

2.3.6 Button id_0 f_screentshot 함수

➢ 브라우저 창에 현재 활성화 된 화면을 캡처를 진행 할 때, 출력 중인 화면이 아닌 해당 url에 해당하는 모든 페이지 화면을 전체 출력한다.

➢ html에서 제공하는 button 중 id_0 식별 값에 연결된 button이며 전체 화면 캡처를 수행하며, 과제에서 원하는 현재화면과 전체화면을 옵션으로 제공하는 부분이다.

```
function f_screentshot (element) {
    if(!!window.openDatabase) {
     alert("현재 브라우저는 Web SQL Database를 지원합니다")
    }
    else{
      alert("현재 브라우저는 Web SQL Database를 지원하지 않습니다")
    }
    chrome.tabs.query({active: true, currentWindow: true}, function(tabs) {
        var tab = tabs[0]; // used in later calls to get tab info
        captureToBlobs(tab)
    });

    var downloadLink = document.querySelector("#MHTML");
    chrome.tabs.getSelected(null, function(tab) {
    chrome.pageCapture.saveAsMHTML({tabId: tab.id}, function (mhtml){
      //alert (tab.url)
      var url = window.URL.createObjectURL(mhtml);
      //alert (url)
      //window.open(url)
      current_download(url, getLocalTimeString()+" "+tab.url.split("://")[1] + ".mht")
      });
    });
}
```

- 현재 브라우저에서 Web SQL DB 지원여부를 확인한다.
- 화면 캡처에 대한 소스코드 저장은 c_screentshot 함수의 진행 절차와 동일하게 획득한다. 해당 소스코드 저장은 보이는 화면 부분의 소스코드가 아닌 활성화 된 url 자체의 소스코드이므로 소스코드 획득은 위와 절차가 동일하다.
- 전체 화면 캡처를 하기 위해 활성화 된 캡처 대상 tab의 정보를 blob 객체로 변환하여 데이터를 다루기 위해 함수를 호출한다. (captureToBlobs)
- Blob : binary large object의 약자, 이진 데이터의 모임이며, 일반적으로 그림, 오디오 또는 기타 멀티미디어 오브젝트를 관리한다.

2.3.7 captureToBlobs 함수

- 활성화 된 tab의 전체 페이지 크기를 탐색하고, 전체 화면이 그림파일로 저장되기 위해 string stream data를 생성하는 함수를 호출하며, string stream data를 그림파일로 변환하는 함수를 호출한다.

```
function captureToBlobs(tab) {
    var loaded = false,
        screenshots = [],
        timeout = 3000,
        timedOut = false;

    chrome.tabs.executeScript(tab.id, {file: 'page.js'}, function() {
        initiateCapture(tab, function() {
            //getBlobs(screenshots)
            full_target_url = tab.url
            getBlobs(full_target_url, screenshots); //blob

        });
    });

    chrome.runtime.onMessage.addListener(function(request, sender, sendResponse) {
        if (request.msg === 'capture') {
            capture(request, screenshots, sendResponse);
            // https://developer.chrome.com/extensions/messaging#simple
            return true;
        } else {
            console.error('Unknown message received from content script: ' + request.msg);
            return false;
        }
    });
}
```

  - 캡처 시작을 위해 initiateCapture 함수에 현재 활성화 된 tab을 가르키는 정보와 getBlobs 함수를 전달한다. 또한 page.js를 현재 main.js에서 실행 시킨다.
  - Chrome.runtim.onMessage api를 이용하여 현재 동작 중에 발생하는 메시지가 있을 경우 동작하도록 하며, capture라는 메시지가 존재하면 capture함수를 실행하도록 한다.

2.3.8 initiateCapture 함수

- scrollPage라는 메시지를 발생시키고, 전달받은 함수를 실행한다.

```
function initiateCapture(tab, call_scroll) {
    chrome.tabs.sendMessage(tab.id, {msg: 'scrollPage'}, function() {
        // We're done taking snapshots of all parts of the window.
        call_scroll();
    });
}
```

2.3.9 getBlobs 함수

- Blob 객체로 변환된 데이터를 image 중 jpeg 형태로 변환한다.
- 변환 과정에서 EXIF 시간 값을 데이터에 밀어 넣은 후, EXIF가 포함된 데이터를 download 함수를 이용하여 사용자의 PC에 저장한다.
- EXIF 삽입 방식과 download 방식은 현재 화면 캡처와 동일하다.

```
function getBlobs(full_url, screenshots) {
    return screenshots.map(function(screenshot) {
        var dataURI = screenshot.canvas.toDataURL('image/jpeg', 0.8);
        blob = full_handleFileSelect(dataURI)
        full_downname = getLocalTimeString()+ " full_("+ full_target_url + ").jpg"
        db_hash = getLocalTimeString()+" "+calcMD5(blob[1]).toUpperCase()+".jpg"
        full_download (blob[0], full_downname)
        InsertData_capture(full_downname, db_hash)
        return blob;
    });
}
```

2.3.10 capture 함수

➢ script 실행 중에 'capture'라는 메시지가 발생되었을 경우 실행 된다.

```
function capture(data, screenshots, sendResponse) {
    chrome.tabs.captureVisibleTab(
        null, {format: 'jpeg', quality: 100}, function(dataURI) {
            if (dataURI) {
                var image = new Image();
                image.onload = function() {
                    data.image = {width: image.width, height: image.height};
                    // given device mode emulation or zooming, we may end up with a different sized image than expected,
                    // so let's adjust to match it!
                    if (data.windowWidth !== image.width) {
                        var scale = image.width / data.windowWidth;
                        data.x *= scale;
                        data.y *= scale;
                        data.totalWidth *= scale;
                        data.totalHeight *= scale;
                    }
                    // lazy initialization of screenshot canvases (since we need to wait for actual image size)
                    if (!screenshots.length) {
                        Array.prototype.push.apply(
                            screenshots,
                            _initScreenshots(data.totalWidth, data.totalHeight)
                        );
                        if (screenshots.length > 1) {
                            $('screenshot-count').innerText = screenshots.length;
                        }
                    }
                    // draw it on matching screenshot canvases
                    _filterScreenshots(
                        data.x, data.y, image.width, image.height, screenshots
                    ).forEach(function(screenshot) {
                        screenshot.ctx.drawImage(
                            image,
                            data.x - screenshot.left,
                            data.y - screenshot.top
                        );
                    });
                    // send back log data for debugging (but keep it truthy to
                    // indicate success)
                    sendResponse(JSON.stringify(data, null, 4) || true);
                };
                image.src = dataURI;
            }
        });
}
```

- 각 설정된 사이즈에 맞게 현재화면과 같은 형식으로 캡처를 하고 주어진 크기에 맞추어 이어 붙이는 형식으로 맨 위부터 아래까지 전체 화면을 완성한다.
- 전체 캔버스를 구성한 후, 저장하는 화면이 생길 때마다 다음 저장 화면의 오프셋을 계산 한 후 변경시킨다.
- 화면 스크린 횟수 계산은 _initScreenshot 함수에서 전체 canvas를 현재 활성화 된 창의 크기에 맞추어 계산한다.
- _filterScreenshots 함수를 이용하여 캡쳐 화면을 canvas에서 지정한 위치에 출력한다.

2.3.11 _initScreenshots 함수

- 최종 이미지의 `totalWidth` 및`totalHeight`을 기반으로 스크린 샷 객체의 배열을 생성한다.
- 너무 큰 경우 Chrome은 이미지를 생성하지 않으므로 여러 캔버스를 고려해야한다.

```
function _initScreenshots(totalWidth, totalHeight) {
    // Create and return an array of screenshot objects based on the `totalWidth` and `totalHeight` of the final image.
    // We have to account for multiple canvases if too large, because Chrome won't generate an image otherwise.

    var MAX_PRIMARY_DIMENSION = 15000 * 2,
        MAX_SECONDARY_DIMENSION = 4000 * 2,
        MAX_AREA = MAX_PRIMARY_DIMENSION * MAX_SECONDARY_DIMENSION;
    var badSize = (totalHeight > MAX_PRIMARY_DIMENSION ||
                   totalWidth > MAX_PRIMARY_DIMENSION ||
                   totalHeight * totalWidth > MAX_AREA),
        biggerWidth = totalWidth > totalHeight,
        maxWidth = (!badSize ? totalWidth :
                    (biggerWidth ? MAX_PRIMARY_DIMENSION : MAX_SECONDARY_DIMENSION)),
        maxHeight = (!badSize ? totalHeight :
                     (biggerWidth ? MAX_SECONDARY_DIMENSION : MAX_PRIMARY_DIMENSION)),
        numCols = Math.ceil(totalWidth / maxWidth),
        numRows = Math.ceil(totalHeight / maxHeight),
        row, col, canvas, left, top;

    var canvasIndex = 0;
    var result = [];

    for (row = 0; row < numRows; row++) {
        for (col = 0; col < numCols; col++) {
            canvas = document.createElement('canvas');
            canvas.width = (col == numCols - 1 ? totalWidth % maxWidth || maxWidth :
                            maxWidth);
            canvas.height = (row == numRows - 1 ? totalHeight % maxHeight || maxHeight :
                             maxHeight);

            left = col * maxWidth;
            top = row * maxHeight;

            result.push({
                canvas: canvas,
                ctx: canvas.getContext('2d'),
                index: canvasIndex,
                left: left,
                right: left + canvas.width,
                top: top,
                bottom: top + canvas.height
            });

            canvasIndex++;
        }
```

2.3.12 __filterScreenshots 함수

- 스크린 샷을 주어진 이미지의 위치와 일치하는 곳에 출력하여 모든 이미지들이 출력 되면 하나의 전체 캡처와 같이 생성한다.

```
function _filterScreenshots(imgLeft, imgTop, imgWidth, imgHeight, screenshots) {
    // Filter down the screenshots to ones that match the location of the given image.
    var imgRight = imgLeft + imgWidth,
        imgBottom = imgTop + imgHeight;
    return screenshots.filter(function(screenshot) {
        return (imgLeft < screenshot.right &&
                imgRight > screenshot.left &&
                imgTop < screenshot.bottom &&
                imgBottom > screenshot.top);
    });
}
```

2.3.13 DB 관련 함수

- Button id_3은 createORloadDB 함수, button id_4 createOrloadTable 함수, button id_5는 showDB 함수에 맵핑 되어, 사용자가 button click시, 해당 함수가 각각 동작하도록 한다.
- DB는 chorme에서 제공하는 Web SQL을 이용하였으며, 해당 DB는 SQLite3의 형식이다.

```
var db;

function createORloadDB(element) {
    if (window.openDatabase) {
        DB_name = dbname.value;
        db = window.openDatabase(DB_name, "1.0", "Capture_Screen", 1024*1024);
    }
}

function createOrloadTable(element) {
    if (!db) {
        alert ("Create Or Load DB First!");
    }
    else {
        db.transaction(function(tx){
            tablequery = "CREATE TABLE " + t_name.value + "(Time,File_Name,Hash)";
            tx.executeSql(tablequery);
        });
    }
}

function DeleteTable() {
    if (!db) {
        alert ("Load DB First")
    }
    else {
        db.transaction(function(tx){
            delquery = "DROP table " + td_name.value;
            tx.executeSql(delquery);
        });
    }
}
```

- createORloadDB 함수 실행 시, web SQL로 DB를 생성 또는 기존에 존재할 경우 불러온다. html에서 제공한 input text box에 사용자가 입력한 string으로 DB 이름을 설정한다
- .createOrloadTable 함수 실행 시, 먼저 DB가 생성되어 있는지 확인을 진행한다. DB가 생성되거나 로드 되었을 경우 해당 DB에 사용자가 입력한 이름으로 Table을 생성하거나 기존 Table을 불러온다.
- Table 생성 시, 함께 생성되는 file명 또한 qurey를 이용하여 지정한다.
- DeleteTable 함수 실행 시, 최초 DB 현황을 확인하고, DB가 선택되어 있을 경우, 사용자 입력에 맞는 Table을 찾아서 삭제한다.

```
function showDB(element) {
    if (!db) {
        alert ("Plz, Set DB & Table First")
    }
    else {
        db.transaction(function(tx){
            showquery = "SELECT * from " + t_name.value;
            tx.executeSql(showquery,[], function(tx,result){
                document.getElementById('print_window').innerHTML = ""
                for(var i = 0; i < result.rows.length; i++){
                    var row = result.rows.item(i);
                    document.getElementById('print_window').innerHTML +=  i+1 + "번\nTime : " + row['Time'] + "\nFilename : "
                     + row['File_Name'] + "\nHash(MD5) : " + row["Hash"] + "\n\n";
                }
            });
        });
        var popUrl = "notice_DB_path.html";//팝업창에 출력될 페이지 URL
        var popOption = "width=843, height=188, resizable=no, scrollbars=no, status=no;";//팝업창 옵션(optoin)
        window.open(popUrl,"Print Web SQL Database",popOption);
    }
}
```

- ShowDB 함수 실행 시, 저장되어 있는 DB 데이터를 textarea에 출력해준다. 사용자가 input box에 선택한 DB와 Table에 있는 정보를 모두 출력한다. (select * from <table>)

➢ DB에 데이터 삽입하는 함수는 InsertData_capture이며, 해당 함수를 이용하여, DB에 채증 시간, 파일 명, 해시 값을 저장한다.

```
function InsertData_capture(filename, hash) {
    db.transaction(function(tx){
        capture_time = filename.split(" ")[0] +" "+filename.split(" ")[1] +" "+ filename.split(" ")[2]
        +" "+ filename.split(" ")[3] + " "+ filename.split(" ")[4];
        hash_value = hash.split(" ")[5].split(".")[0]
        Inquery = "INSERT INTO " + t_name.value + "(Time,File_Name,Hash) values(?,?,?)"
        tx.executeSql(Inquery,[capture_time, filename, hash_value]);
    });
}
```

- 데이터를 저장할 DB와 Table 활성화 여부를 확인한다.
- 채증 시간, 파일명, 해시 값을 DB에 넣기 위해서 먼저 filename, hash라는 변수로 데이터를 받아온다. 채증하여 파일을 다운로드 할 때, 지정하는 파일명과, md5를 생성하는 알고리즘을 이용하여 string stream data를 암호화 한 값을 각각의 변수명에 매칭하여 받아온다.
- Filename, hash라는 변수명에 있는 값들을 원하는 부분만 사용하도록 자르거나 이어붙여서 각 table의 필드에 맞게 넣는 query를 생성한다.

2.4 Page.js

➢ Main.js에서 page.js를 excutescript로 실행한 이유는 initiateCapture에서 전송하는 "scrollPage' 라는 메시지를 전달받아 수행할 수 있는 함수가 존재하기 때문이다.

➢ Page.js의 함수는 전체 페이지 오프셋을 확인하여 좌표를 이동시키면서 scroll이 가능하도록 한다.

➢ Page.js는 scroll을 하면서 얻은 전체 크기를 이용하여 각 부분 사진을 찍기 위해 cap[ture라는 메시지를 발생 시키는 역할도 있다.

```
function onMessage(data, sender, callback) {
    if (data.msg === 'scrollPage') {
        getPositions(callback);
        return true;
    } else if (data.msg == 'logMessage') {
        console.log('[POPUP LOG]', data.data);
    } else {
        console.error('Unknown message received from background: ' + data.msg);
    }
}
```

- 전송 받은 메시지가 'scrollPage'의 여부를 확인하고, page.js의 getPosition 함수를 수행한다.

```
var data = {
    msg: 'capture',
    x: window.scrollX,
    y: window.scrollY,
    complete: (numArrangements-arrangements.length)/numArrangements,
    windowWidth: windowWidth,
    totalWidth: fullWidth,
    totalHeight: fullHeight,
    devicePixelRatio: window.devicePixelRatio
};
```

- 계산된 길이를 이용하여 data struct를 구성하고 capture 메시지를 발생시켜 화면 캡처를 수행하도록 한다.

- https://github.com/BoBpromoto/WebAcquisition_Tech_04/blob/master/page.js

2.5 piexif.js

- JPG에는 EXIF 식별 정보가 포함 가능하다. EXIF의 형식에 맞게 Tag와 값 입력이 가능하도록 설정된 javascript이다.
- 과제에서 EXIF에 별도의 표준시 정보를 포함하는 조건 충족을 위해 해당 javascript의 요소들을 호출하여 사용한다.

2.6 Notice_DB_pathhtml & style2.css

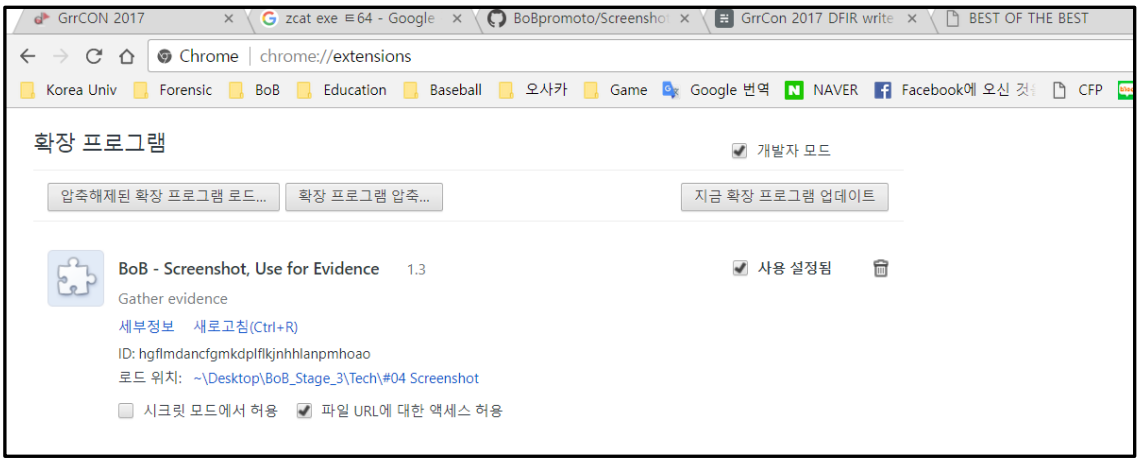
- 저장된 DB 파일이 있는 경로를 popup창으로 출력하여 사용자에게 보여준다.
- popup창으로 출력하여 사용자의 주의를 환기시키는 역할을 하고, popup창의 내용을 숙지하여 사용자 PC내의 DB에 접근하고 사용자 관리가 용의하도록 한다.
- 2.3.13항의 2번째 사진을 참조하면 showDB에서 새로운 창을 실행하는 부분이 존재한다.
- DB 데이터 출력과 동시에 popup창을 출력한다.

```
<!DOCTYPE html>
<html><head lang="en">
  <meta charset="UTF-8">
  <title>Website Screenshot</title>
  <link href="http://fonts.googleapis.com/css?family=Open+Sans:400,700" rel="stylesheet" type="text/css">
  <link href="style2.css" rel="stylesheet">
</head>
<!-- <script type="text/javascript" src="./main.js"></script> -->
<!-- <script type="text/javascript" src="./printdb.js"></script> -->
<div class="window_header">
  <h2>Browser Screenshot</h2>
  <p>Best of The Best 6th Digital Forensic</p>
</div>
<div class="path">
  <h2> Default DataBase Path : C:\Users\(username)\AppData\Local\Google\Chrome\User Data\Default\databases\chrome-extension_...</h2>
  <p> if not exist in defalut path, check your DB path. => input url : chrome://version and check profile path</p>
  <a> DB File Format = SQLite Format 3</a>
</div>
</body>
</html>
```

- DB 경로 표시 창에 대한 style 적용은 style2.css를 이용하여 관리하며, 동작 원리는 앞서 언급한 style.css와 동일하다.

3. 확장 프로그램

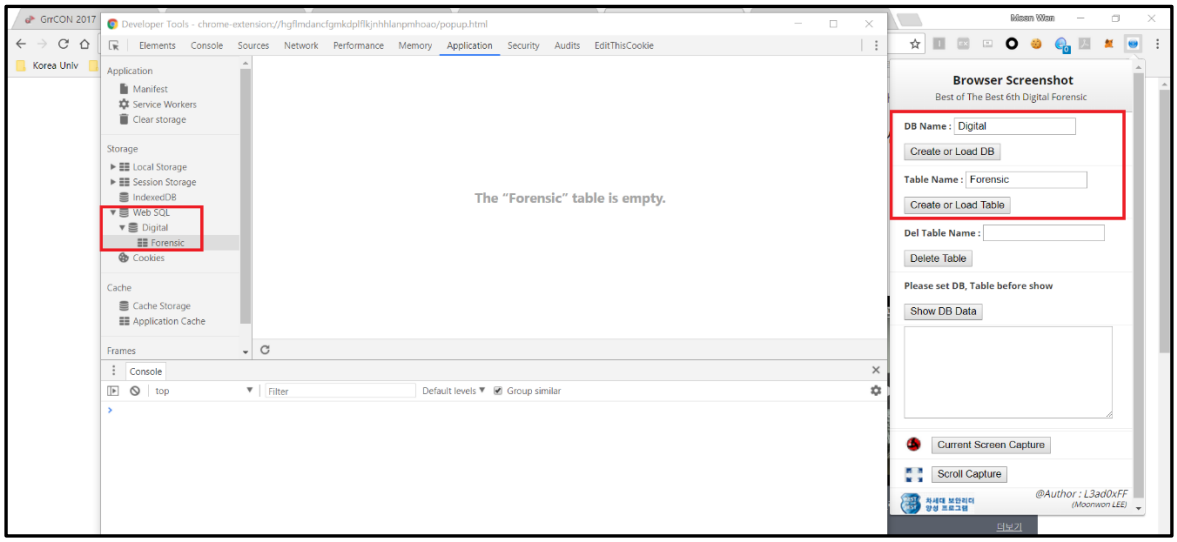
3.1 확장 프로그램 등록



➢ Chrome://extension 에서 작성한 확장 프로그램을 등록한다.

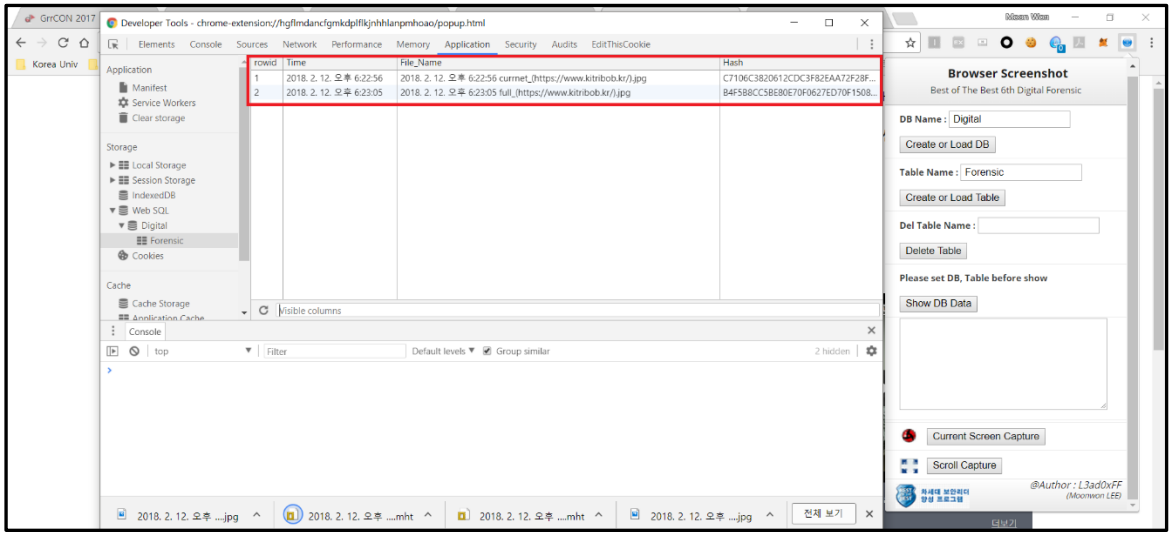
3.2 확장 프로그램 실행

➢ 확장 프로그램이 등록되면 주소 창 오른편에 설정한 icon이 생성되고 해당 icon을 click하면 위와 같은 popup.html이 실행된다.

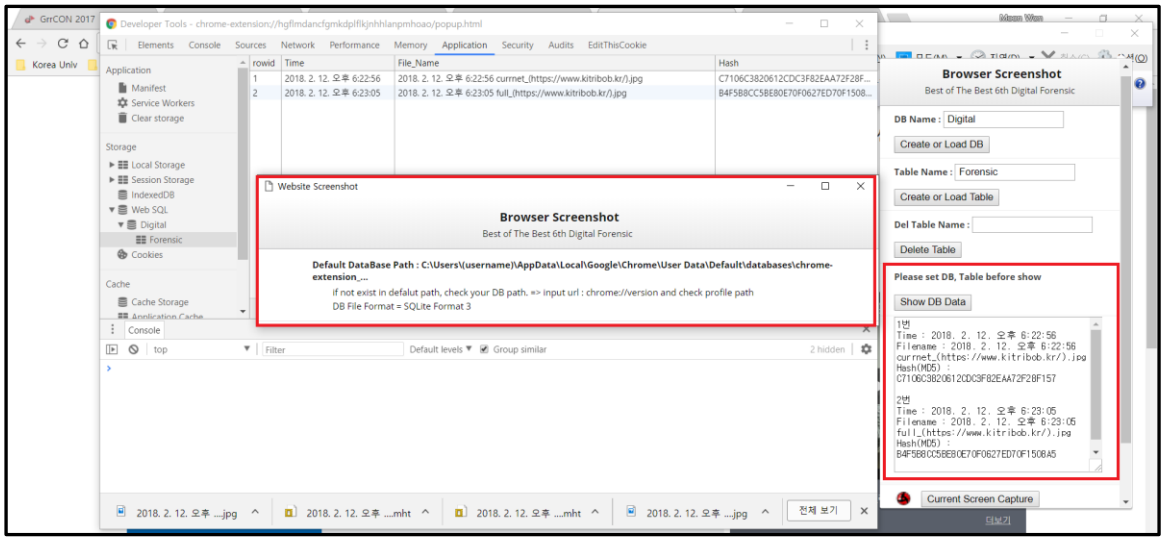


➢ 사용자의 입력에 따라 단계별로 버튼을 누르면 Web SQL에서 DB와 Table이 생성됨을 확인 할 수 있다.

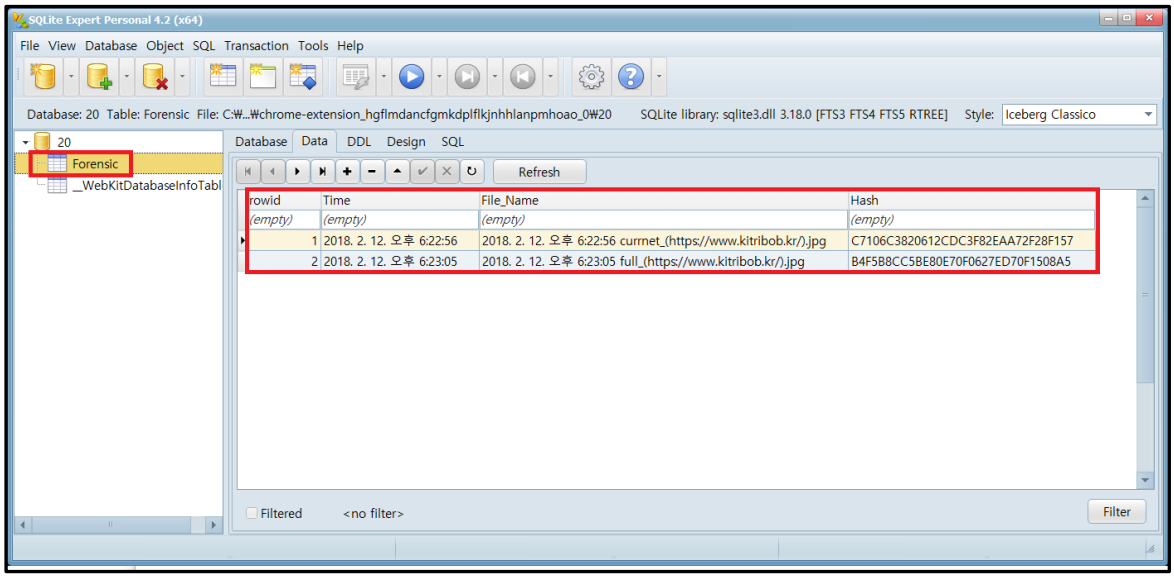
➢ 확장 프로그램 실행 시, drop형태로 제공되는 html 파일에서 F12 또는 오른쪽 마우스 클릭 후 검사 항목을 누른 후, Application 항목에서 확인 할 수 있다.



➢ 현재 화면 캡처와 전체화면 캡처를 순차적으로 실행하면, 하단 표시줄에 다운로드 파일들과 각 파일들로부터 저장할 데이터들이 DB에 저장됨을 확인 할 수 있다.



➢ DB내의 데이터를 조회하면 사용자가 입력한 DB와 Table에 저장된 정보를 출력과 동시에 새로운 pop창을 제공하여 저장된 경로를 출력한다.



➢ 해당 경로로 이동하여 생성된 DB File을 SQLite 도구를 이용하여 열어본 결과이다. 확장 프로그램을 이용하여 생성한 DB의 정보가 존재한다.

DB의 경로는 default로 chrome에서 지정한 경로에 저장이 된다. 경로의 경우 chrome의 버전마다 상이할 가능성도 존재하기 때문에 'chrome://version'에서 profile paht를 확인이 필요하다.

기본적인 경로는 C:₩Users₩(username)₩AppData₩Local₩Google₩Chrome₩User Data₩Default₩databases₩<chrome - 확장자 생성 시, 자신의 id>.

4. 참고문헌

- Google API : https://developer.chrome.com/apps/api_index
- Full Screenshot javascript : https://github.com/mrcoles/full-page-screen-capture-chrome-extension
- Insert EXIF Tag value : https://github.com/hMatoba/piexifjs
- Chrome Web SQL : https://developers.google.com/web/tools/chrome-devtools/manage-data/local-storage?hl=ko