Archival Acid Test

Master’s Project Final Report

Author: Hung Do
Email: hdo@odu.edu

Project Advisors:

Dr. Michele Weigle
Email: mweigle@cs.odu.edu

Dr. Michael L. Nelson
Email: mln@cs.odu.edu

Project Presentation Date: April 27, 2017

Department of Computer Science
Old Dominion University
# Table of Contents

Acknowledgment 3

1. Introduction 4

2. Archiving Process 6

3. Archival Acid Test 7
   - Set 1: Acid Test on Image and CSS 9
   - Set 2: Acid Test on Javascript 10
   - Set 3: Acid Test with HTML5 Tags 13
   - Set 4: Testing on Delay Embedded Resource 15
   - Set 5: Acid Test on Leakage Resource 16
   - Set 6: Content Security Policy 17

6. Evaluation 19

4. Future Work 21

Conclusion 21

References 22

Resources 22
Acknowledgment

I would like to thank Dr. Michele C. Weigle, Associate Professor and Dr. Michael L. Nelson, Professor from Web Science and Digital Library Research Group of the Computer Science at Old Dominion University who giving many insightful advices and valuable inputs. The guidance from you is extremely important for the success of the project.

I would also like to thank all other members from Research Group who providing related resources and important comments during my work with the Group. My special thank to Mat Kelly, PhD Candidate who has done initial work of this project.
1. Introduction

The function of web archiving is to preserve web pages that exist on the live web so that user can replay them later regardless of the change or availability of the live web. A typical archive tool also provides the ability to store different snapshots of a live webpage at different times so that user will be able to view web pages from the past in the Wayback Machine.

However, due to the complexity of web page structure, archival tools face challenges when preserving the web page. Many examples present these challenges such as recursive load of Javascript and/or CSS, these challenges may lead to low quality of web archives.

Web preservation process

Process of building an archive including two operations. First, archival tool will reach out to the live-web and download the current page with all related resources for the page including images, videos, javascripts, and etc. Secondly, it rewrites resource URLs that are downloaded in the previous step and stores the content in an independent document, usually the document is in the WARC (Web ARChive) format. Below is an example of an image url (resource) has been rewritten upon replay. The rewritten url has a prefix added in order to use the archived resource when replaying.

From live web:

```html
<img src="http://www.cs.odu.edu/~hdo/acidtest/images/pixel.png" title="test1b" id="test1b"/>
```
Acid Test is designed and focused on issues and challenges of preservation specifically, by this way, reduce the complexity when judging capability of archival tool. In more detail, we test an archival tool based on ability of preserving resources such as images, javascripts, page structure, potential leakage and how the archive will be replayed in browser.

Figure 1.1 is an example\(^1\) of the damage of archives. This was archived in 2008 and on the left is the campaign presidential campaign between McCain-Palin and Obama-Biden. However due to a javascript leakage, when this archived was observed in 2012 it displays an advertisements on 2012 presidential race between Romney-Ryan and Obama-Biden.

\[\text{Figure 1.1}\]

That is why we need a method to test these challenges for web archive tool. We created an Acid Test Page that contains a set of tests to simulate challenges that impact the quality of web archives. The main purpose of Acid Test is to evaluate an archival tool by archiving the Acid

\(^1\)Example taken from blog by Dr. Justin F. Brunelle: http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html
Test Page from that tool. Acid Test will score the performance of the archival tool based on these test results.

2. Archiving Process

In this section we will discuss about the process of archiving and how an archive is replayed and presented the content in browser.

Process of archiving

Figure 2.1 is a typical process of archiving. There are three main entities in this diagram Browser Web Archive and Web Server. Usually, we use a browser to open a Web Archive and input an URL to live web which we need to archive its content. Web Archive will do a request to Web Server via the URL then stores the result into database.

![Figure 2.1]
Archival tools

In addition, different archival tool has different approach when archiving a web page. This project used three popular archival services during test development those are archive.Is, webrecorder.io and web.archive.org. Each of these have different approach and each one has pros and cons.

Rewrite upon replay

Typically, all urls in an archive must have been rewritten in order to make sure all resource replay independently to the live web. This requirement creates challenge for archiving process on how to find and rewrite urls appropriately. Acid Test has a specific test for this challenge in Set 5: “Acid Test on Leakage Resource” which will be discussed in detail later.

3. Archival Acid Test

This section will discuss the detail of each test in the Acid Test Page, why it has been included and how to use it.

How to use this Acid Test

Use this Acid Test homepage (http://www.cs.odu.edu/~hdo/acidtest/) as an input for an archival tool that needs to be evaluated. It takes about 10 seconds for loading the Acid Test page. After that, there is a Total Score display once the Acid Test page is loaded completely. Below is the screenshot of the Acid Test page (include first set of the tests) in webrecorder.io. When loading complete the Total Score will be displayed.

2 GitHub repo of this project: https://github.com/hungvietdo/archivalacidtest
Design of the Tests

This Acid Test is divided into five categories. The tests are focus on areas that an archival tool might be failed to capture or re-write appropriately into memento leading to incorrect representation when playing back.
Test from set 1 and 2 have been developed by Mat Kelly\(^3\), in this project these tests have been reused with added feature on scoring for each test, by this way we can consolidate a total score for a particular archival tool.

**Set 1: Acid Test on Image and CSS**

Focus on image and css preservation with different way of setup from each detail test such as relative local image, absolute local image, or recursively included CSS, and etc.

Figure 3.2 is what we will see from the Acid Test page for tests in set 1.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1a</td>
<td>Local Image, relative</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 1b</td>
<td>Local Image, absolute</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 1c</td>
<td>Remote Image, absolute</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 1d</td>
<td>Inline Content, Encoded Image</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 1e</td>
<td>Scheme-less resource</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 1f</td>
<td>Recursively included CSS</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>6 of 6</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.2**

**Test 1a:** The image is a local image, if an archival tool can archive the image then a blue rectangle will show up in status column.

\[
\text{<img src="images/pixel.png" title="test1a" id="test1a"/>}
\]

\(^3\) More detail about Mat’s project can be found at this: [http://acid.matkelly.com/](http://acid.matkelly.com/)
**Test 1b:** With an absolute image source.

```
<img src="http://www.cs.odu.edu/~hdo/acidtest/images/pixel.png" title="test1b" id="test1b"/>
```

**Test 1c:** Same image as above two tests but in a different domain with the domain of Acid Test

```
<img src="https://raw.githubusercontent.com/hungvietdo/archivalacidtest/master/images/pixel.png" title="test1c" id="test1c"/>
```

**Test 1d:** Use an encoded as image source and attach directly to the image tag.

```
<img src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADAAAAAbAIAAAACQ1DeAAAAGXRFBWHRTbZ6d2FyZQBBG9jZSBbWFnZVJhYWR5cmlpbwAAAABJRU5ErkJggg==" title="test1d" id="test1d"/>
```

**Test 1e:** Using a schemaless URL, without http nor https in URL

```
<img src="/images/pixel.png" title="test1e" id="test1e"/>
```

**Test 1f:**
In this test we will test the CSS calls from multiple CSS files. In order to display an archive correctly, archival tool should be able to archive all the CSS files.

First css file:

```
@import url("1f_recursive.css");
```

Second css file: (called by the first css file)

```
#onef {background-image: url('images/pixel_1f.png')};
```
Set 2: Acid Test on Javascript

This set of test are focus on testing the preservation of javascripts. Acid Test uses several types of JavaScript to test the behavior such as local, remote, inline AJAX content.

Figure 3.3 shows the tests in set 2. If archival tool fails one test the the status column will show up a red rectangle and mark score as zero.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 2a</td>
<td>Script, local</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2b</td>
<td>Script, remote</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2c</td>
<td>Script inline, DOM Manipulation</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2d</td>
<td>AJAX image replacement of content</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2e</td>
<td>Code that dynamically adds styleheets</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 2f</td>
<td>Code that manipulates DOM after a certain delay (test the synchronicity of the tools)</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.3

**Test 2a:** A local javascript will be added. If archival test will be able to run this script successfully then a blue rectangle image will show in the status column.

```html
<script type="text/javascript" src="js/localScript.js" id="test2a"></script>
```

The localScript.js will generate image and add to status column above:

```javascript
document.addEventListener('DOMContentLoaded',function(){
    var localScriptImage = new Image();
    localScriptImage.src = "images/pixel.png";
    localScriptImage.title = "test2a";
    var scriptParent = document.getElementById('scriptParent_2a');
    scriptParent.replaceChild(localScriptImage,document.getElementById('test2a'));
});
```
Test 2b: In this test will call a javascript from a different domain with the domain of Acid Test page. Archival tool must be able to archive this script to replay the archive correctly.

```html
<script type="text/javascript" src="https://rawgit.com/hungvietdo/archivalacidtest/master/js/externalScript.js" id="test2b"></script>
```

Test 2c: This javascript will create a blue rectangle image and append directly to status column for test2c.

```html
<script type="text/javascript">
// Test 2c, Script inline, DOM Manipulation
document.addEventListener('DOMContentLoaded', function(){
  var inlineScriptImage = new Image();
  inlineScriptImage.src = "pixel.png";
  inlineScriptImage.title = "test2c";
  var scriptParent = document.getElementById('scriptParent_2c');
  scriptParent.appendChild(inlineScriptImage);
});
</script>
```

Test 2d: In this test will use ajax to load image and check the web archival tool to see if it can load and display the image when replaying the web archive.

```html
<script type="text/javascript">
document.addEventListener('DOMContentLoaded', function(){
  // Test 2d init AJAX
  var xhr = new XMLHttpRequest();
  xhr.open('GET', 'images/pixel.png', true);
  xhr.responseType = 'arraybuffer';
  xhr.onload = function(e) {
    document.getElementById('test2d').src = "data:image/png;base64,"+base64ArrayBuffer(e.currentTarget.response);
  }
  xhr.send();
});
</script>
```
**Test 2e:** A dynamic stylesheet will be added and modify the display of the image. Archival tool should be able to store this and modify the display appropriately.

```html
<script type="text/javascript">
// Test 2e: Dynamically added stylesheets
document.addEventListener('DOMContentLoaded', function(){
  // Test 2e setup
  var linkTag = document.createElement('link');
  linkTag.href = "css/2e.css";
  linkTag.rel = "stylesheet";
  linkTag.type = "text/css";
  document.head.appendChild(linkTag);
});</script>
```

**Test 2f:** This test will delay 5 seconds before set a blue rectangle image to the status column.

```html
<script type="text/javascript">
document.addEventListener('DOMContentLoaded', function(){
  // change image after 5 seconds
  setTimeout(function(){
    document.getElementById('test2f').src = "images/pixel.png";
  },5000);
});</script>
```

**Set 3: Acid Test with HTML5 Tags**

These tests focus on some new features of HTML5 such as canvas, local storage, iframe with sandbox tag, and video tag. Acid Test will use these test to evaluate how well archival tool can do on these new HTML5 tags.

There are four tests showing in:

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 3a</td>
<td>HTML5 Canvas Drawing</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 3b</td>
<td>HTML5 Local Storage</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 3c</td>
<td>New HTML5 sandbox tag</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Test 3d</td>
<td>HTML5 video tag</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Video courtesy of Big Buck Bunny (www.bigbuckbunny.org).

Total 4 of 4
**Figure 3.4**

**Test 3a:** In this canvas test, if a canvas element can be created successfully then a blue rectangle image will be displayed in status column (see above screenshot).

```javascript
var elem = document.createElement('canvas');
if (elem.getContext & elem.getContext('2d')) {
    document.getElementById('test3a').src = 'images/pixel.png';
}
```

**Test 3b:** In this test, an item will be added to local storage, a red rectangle will show up if this test is failed.

```javascript
var test = 'test';
try {
    localStorage.setItem(test, test);
    localStorage.removeItem(test);
    document.getElementById('test3b').src = 'images/pixel.png';
} catch(e) {
    document.getElementById('test3b').src = 'images/red.png';
}
```

**Test 3c:** Sandbox tag in html5 allows an iframe to define restrictions for the content of the iframe. In this test we will allow script in the iframe and that script can only come from same domain with the test. Below is the iframe with sandbox tag:

```html
<iframe id="iframe" height="60" sandbox="allow-scripts allow-same-origin"></iframe>
```

To validate the iframe with sandbox tag, we setup a script below, if this script run successfully then the status of this test will change to a blue rectangle image.

```javascript
var iframeDoc = document.getElementById('iframe').contentWindow.document;
if ($(iframeDoc).find('#iframe text').text() == 'New HTML5 sandbox tag') {
    document.getElementById('test3c').src = 'images/pixel.png';
}
```
**Test 3d:** This test will load a short video and try to play it. If archival tool is not able to archive the video appropriately then there will be an error and eventually

```javascript
var canPlay = false;
$.ajax({
  url: 'video/movie.mp4',
  async: false,
  error: function()
  {
    // file does not exists
    document.getElementById('video_status').innerHTML = 'Can not play the movie.';
  },
  success: function()
  {
    // file exists
    document.getElementById('test3d').src = 'images/pixel.png';
    var v = document.createElement('video');
    if (v.canPlayType && v.canPlayType('video/mp4').replace(/no/, '')
    {
      document.getElementById('test3d').src = 'images/red.png';
    }
  }
});
```

**Set 4: Testing on Delay Embedded Resource**

Browser usually makes several requests to webserver, sometimes webserver has slowness when sending responses back. In particular, when archiving, an archive tool will play as a browser to send request to web-browser. If webserver is slow in sending response, archive tool may be exhausted and consider the resource is not available. This can lead to a low quality of the archive.

Figure 3.5 is an example from Internet Web Archive when it archives a resource that take a little longer in response (8 seconds) then this archive tool considers the unavailability of the resource.
This test is setup as five independent images with different response time 1, 2, 4, 6, 8 seconds respectively. Each of these image represents for an embedded resource with delay. This test will try to measure how long that a crawler can wait.

If archive tool is not able to wait for all these resources we will see a result as below:

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 4</td>
<td>Delay on embedded resource</td>
<td>12468</td>
<td>Can wait response for 4 seconds, but not for 6 seconds.</td>
</tr>
</tbody>
</table>

Otherwise, we will see a result as below:

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 4</td>
<td>Delay on embedded resource</td>
<td>12468</td>
<td>Can wait at least 8 seconds.</td>
</tr>
</tbody>
</table>
Set 5: Acid Test on Leakage Resource

Zombie in the Archive

This test is designed to check if an archival tool will be able to preserve resource that is dynamic or hide in JavaScript, For example: a javascript function composes an image url and web archival tool is not able to recognize this resource to rewrite appropriately. Not be able to do so will lead to a leakage resource issue, this is an important feature for any archival tools as discuss in the previous paragraph.

**Test design:** Two images are created automatically on live web depend on the time of page loading. The tricky part of this test is to hide the url inside JavaScript (this is actually have number of time happened in the live web).

If an archival tool is unable to preserve the later image then it fails in this test because of the leakage resource.

See the Figure 3.6 for the test, these two images are identical from live web. Archival tool fails on this test will show different datetime in the images.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Content</th>
<th>Status</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 5</td>
<td>Relative image</td>
<td>2017-03-27 01:02:31+09:00 UTC</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Absolute url from JavaScript</td>
<td>2017-03-27 01:02:31+09:00 UTC</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.6**

---

4 Blog by Dr. Justin F. Brunelle: http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html
Set 6: Content Security Policy

“Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS)”\(^5\) This is the main reason why CSP has been using widely in live web.

There are two main places that CSP can be implemented either in the Header and/or Metadata Tag. Below are examples how to define CSP in PHP and Metadata Tag.

**PHP:**

```php
```

**Metadata:**

```html
<meta http-equiv="Content-Security-Policy" content="img-src 'self'; script-src 'self'">
```

In PHP, this rule allows only images and scripts from same domain of the page. No other outside resources (images and scripts) can be used. In Metadata as below have same effect as above, only allow images and scripts from same domain. In the case CSP is defined in both header and metadata tag, browser will intersect these two set of rules when rendering the page.

This feature is efficient for live web to control and secure the web representation, however it creates unexpected results and challenges for archival tools when preserving and replaying mementos that have Content Security Policy.

Example in figures 3.7 and 3.8 shows an example of a memento has Content Security Policy implementation and it leads to an unexpected difference between live web and memento.

Figure 3.7: Live web with CSP that does not allow displaying of second image with CSP defined in the header:

```
```

Figure 3.7

Header was modified by web archival tool and will be ignored by browser.

HTTP/1.1 200 OK
Server: Tengim/2.1.0
Date: Fri, 28 Apr 2017 04:43:19 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 3308
Connection: keep-alive
Content-Encoding: gzip
Accept-Encoding: gzip, deflate

Figure 3.8: the archive of this page displays both images, at this point browser ignore the CSP definition inside memento.

Figure 3.8
4. Evaluation

How well an archival tool performs when preserving a web page.

Each test will be marked as a blue rectangle in status column if it does well on that test. Otherwise a red rectangle will be displayed which indicates a fail on that test.

A passed test has score value as one. Total of these score will be display in the beginning when Acid Test page has been loaded completely.

Tests in part 4 and part 5 have no score since they are designed to test the ability of waiting from embedded resource and a checking on leakage behavior if any.

Among three archival tools archive.is, webrecorder.io and web.archive.org, webrecorder.is has passed all the tests. Web.archive.org fails on the leakage test and can only wait for five seconds for resource response. Archive.is fails on html5 video tag.

Score for webrecorder.is

![ Acid Test Result for webrecorder.is ]

Archive this page from an archival tool to get the result.

Total score is **17/17**
Score for archive.is

Score for web.archive.org

web.archive.org fails on leakage test.

The archive is from April 02, 2017 but when replaying on April 25, 2017 the second image displayed the current date since this is a leakage resource (browser gets content directly from liveweb not from archive).
5. Future Work

Due to the time constraints, several potential tests have not been completed and included into the current project. One of those is CSP tests. Challenge of adding CSP into Acid Test page is the policies from CSP will be applied directly to Acid Test page and these policies will affect negatively to other tests of Acid Test.

Also as the nature of changing in web technology such as HTML5 there will be more potential tests can be added to the Acid Test in the future.

6. Conclusion

Acid Test is created to quantify it in an efficient way since this Acid Test is an independent web site with pure nature and focus only on those area that may be challenge for web archival tool. Just by loading Acid Test Page we will quickly see the overall of an archival tool.

In this project, I have expanded all tests from previous project by Mat Kelly and added HTM5, delay on resource and leakage tests. All the tests can be measured by score as well as total score for the whole Acid Test.
References

