Summer 2012

An Extensible Framework for Creating Personal Archives of Web Resources Requiring Authentication

Matthew Ryan Kelly
Old Dominion University

AN EXTENSIBLE FRAMEWORK FOR CREATING PERSONAL ARCHIVES OF WEB RESOURCES REQUIRING AUTHENTICATION

by

Matthew Ryan Kelly
B.S. June 2006, University of Florida

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2012

Approved by:

_____
Michele C. Weigle (Director)

_____
Michael L. Nelson (Member)

_____
Yaohang Li (Member)

ABSTRACT

AN EXTENSIBLE FRAMEWORK FOR CREATING PERSONAL ARCHIVES OF WEB RESOURCES REQUIRING AUTHENTICATION

Matthew Ryan Kelly
Old Dominion University, 2012
Director: Dr. Michele C. Weigle

The key factors for the success of the World Wide Web are its large size and the lack of a centralized control over its contents. In recent years, many advances have been made in preserving web content but much of this content (namely, social media content) was not archived, or still to this day is not being archived, for various reasons. Tools built to accomplish this frequently break because of the dynamic structure of social media websites. Because many social media websites exhibit a commonality in hierarchy of the content, it would be worthwhile to setup a means to reference this hierarchy for tools to leverage and become adaptive as the target websites evolve. As relying on the service to provide this means is problematic in the context of archiving, we can surmise that the only way to assure that all of these shortcomings are not experienced is to rely on the original context in which the user views the content, i.e. the web browser. In this thesis I will describe an abstract specification and concrete implementations of the specification that allow tools to leverage the context of the web browser to capture content into personal web archives. These tools will then be able to accomplish personal web archiving in a way that makes them more robust. As evaluation, I will make a change in the hierarchy of a synthetic social media website and its respective specification. Then, I will show that an adapted tool, using the specification, continues to function and is able to archive the social media website.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Michele C. Weigle, for her guidance, support, and facilitation of an idea that otherwise would have never come to fruition. I would also like to thank my committee for their input and for keeping the document down to a sane scope. Finally, I would like to thank my wife, Melissa, for reasons that would exceed the length of this document if enumerated in a very small font.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The key factors for the success of the World Wide Web are its large size and the lack of a centralized control over its contents [13]. Web crawlers were created to traverse the web and collect information for indexing by search engines so that the information contained in the pages on the web could be found. Through this indexing, a webpage is considered "surfaced" from the Deep Web [8] (that is, the set of all pages not accessible through search engines). Even when the content disappears, the reference can continue to exist, leaving only a remnant of a resource that once was. Estimates for the average lifetime of a web page vary and include concrete estimates, such as 44 days, and approximations based on the size and location of the content itself [16, 23, 30]. Many believe that if you find something on the Internet once, it will be there when you look for it again, suggesting an almost magical persistence [45]. But, this is not the case and oftentimes data is not preserved. The original notion of the web crawler as an indexing tool has been extended to account for the laborious task of preserving as much of what can be considered an infinite amount of content [13] in a form that would allow the content to be re-experienced (i.e. replayed) to those that may have not been aware of the content when it originally existed on the web.

In recent years, many advances have been made in preserving web content. Efforts from organizations like the Internet Archive[1] have preserved content on the web that would otherwise be lost in time. The Internet Archive, in particular, has provided end-user access to its archives through the web-based Wayback Machine[2]. While the Internet Archive has grown in the extent of the web it archives (about 12 petabytes as of early 2012 [12]) since its inception in 1996 [29], there is much that has been lost in time [7]. Much of this content was not archived, or still to this day is not being archived, for reasons of content quality, the assumption that the web is self-preserving, or the belief that archiving is not possible [48].

[1] http://www.archive.org
[2] http://archive.org/web/web.php

Information distributed on the web encompasses a vast array of the activities and artifacts of humanity [19]. One very important area of the web that contains much information, yet is not being preserved, is the user-generated content on social media websites. This is largely because this content resides behind authentication. Archiving services and tools do not currently attempt to capture this content because of its reliance on a context inapplicable to the crawler. Content viewed by a user who has authenticated with the targeted service is often tailored to a user's history, relationships, and other factors that do not pertain or are not appropriate from the perspective of a web crawler. Relying solely on lack of context by a web crawling service as justification for not preserving this content is an insufficient reason to prevent this content from being preserved. Instead, the source, actions, and disposition of the content should be considered [41], and the intrinsic value of such content should be realized as content on these websites becomes a larger cornerstone of individuals' respective digital histories. Individuals who would like to archive this content are currently unable to do so. However, providing the means is achievable by translating the task performed by the crawler to the context of the user.

In this thesis I will describe an abstract specification and concrete implementations of the specification that allow tools to leverage the context of the web browser to capture content into personal web archives. These tools will then be able to accomplish personal web archiving in a way that makes them more robust. As evaluation, I will make a change in the hierarchy of a synthetic social media website and its respective specification. Then, I will show that an adapted tool, using the specification, continues to function and is able to archive the social media website.

I.1 PROBLEM

Social media websites, by their very nature, are extremely dynamic in regards to design, content and offerings. Services that exhibit this trait frequently provide the facilities (e.g., an API) for a user or third party to utilize the content contained on the website without having to query the service's databases directly. This allows the service to control what content is released to those that utilize the API but at least provides a mean for users to acquire this otherwise protected content.

Relying on service-provided APIs, however, leaves open the potential for exclusion or manipulation [53] of content at the discretion of the service, lack of look-of-feel preservation of archived content when comparing that to be archived versus the

archived content, and breaking of code that utilizes the API when a certain feature is disabled, deprecated or otherwise modified[3]. Even if the APIs give adequate access to data, the complex and ever-changing terms of use[4], permissions policies and individual privacy preferences make archiving a considerable, even well-nigh impossible challenge [68]. From this, we can surmise that the only way to assure that all of these shortcomings are not experienced is to rely on the original context in which the user views the content, i.e. the web browser.

Because web browsers are meant to play content fetched and not preserve the content, some programmatic approach must be taken to extend the browser to provide this additional functionality. All modern browsers provide some facility to extend the browser through an "extension" or "add-on" sub-system with an API that provides access to the browser's core functionality. Developers can create tools that leverage the browser context, which frees them from coding against the wide variety of platforms and instead allows them to utilize a single browser API. Tools have been created that allow a user to not be restricted by the data obtained through service-level APIs and instead deliver a direct means of providing the preservation ability that browsers have previously lacked. Projects like the Mozilla Firefox add-on Archive Facebook [51] attempt to enable users to save this content (in this case, a user's data on facebook.com), but such tools are prone to break because of the dynamic nature of the target websites. Further, such tools' procedures are likely to cease functioning or will function incorrectly because of their reliance on scraping and regular expression based parsing schemes.

Archive Facebook's name is a slight misnomer because its output has greater similarity to a backup than an archive. The tool is capable of capturing the content on a user's Facebook page into a local directory accessible (consisting of the HTML and images file that represent the user's Facebook content) from the web browser. However, the user is unable to relocate the "archive" to ensure preservation, extensive information about the capture procedure is not retained and the data is not in a self-contained format that would facilitate archive cohesion. From the perspective of the Internet Archive's Wayback Machine as an end-user, the output of Archive Facebook seems sufficient, but the input to the Wayback Machine is more technical, structured

[3]An example can be seen in the frequent deprecation of API features by Twitter at https://dev.twitter.com/docs/deprecations/spring-2012

[4]Yahoo! Search Boss API http://developer.yahoo.com/search/boss/ once offered unlimited free use but now charges on a basis of number of queries performed

and general purpose than the add-on's backup. The ultimate objective of Archive Facebook is to allow a user to preserve the content that resides on Facebook, which some claim to be its users' scrapbook, yearbook and Guinness World Record [37].

To be more like a standard archive, Archive Facebook would need to produce output in a more suitable format like that produced by Internet Archive's Heritrix crawler and consumed by Wayback Machine: the Web ARChive (WARC) format [35]. WARC gives structure, standardization and motivation for preserving content in a way beyond backup procedures like Archive Facebook's. As few tools beyond the Wayback Machine utilize the WARC format, the potential to which it might have in further advancing developments in personal web archiving are numerous.[5]

I.2 APPROACH

The aim of this research is to explore the hierarchy of social media websites and provide methods, means, and direction for personal web archiving. The structure of the section breakup of various social media websites will be investigated and a resulting class structure to generalize these sections into a class-like hierarchy will be proposed. This hierarchy will also introduce the abstraction of actions to be performed on these sections to facilitate comprehensive archiving. Once this structure is established, a schema will be extrapolated to represent the section structure in a usable and standardized format to be consumed by services that wish to leverage the specification. The merits and downsides of using a standard specification in this way, especially relating to the ephemerality of social media websites' structure and design, will be discussed and the problems addressed. Sample implementations of the specification will also be described and provided to show that application of the standard is accessible and can be applied to current tools so that these tools can benefit from the advances that this standard provides. The specification will be composed in a way to encourage extension and increased applicability to types of social websites that do not currently exist. Creating a specification with its target being the implementation by software projects is a design task. The specification is progressively built in this thesis with a rationale being supplied for each element in the specification. Allusion to the respective section from which the element is derived is also supplied to validate that the element has a practical application. The structure

[5]One particular example of a use case that requires content to be archived in a standardized format is in the Memento [69] project, which allows one to easily traverse websites over time.

is verbose in its nomenclature so as to be explicitly semantic and to be extensible in an intuitive way when it is appended or otherwise modified as the target social websites evolve and new ones with innovative site hierarchies come into existence.

Following the definition of the specification, and particularly the concrete definition of Facebook's hierarchy, Archive Facebook is re-programmed to conform to the specification so as to be adaptive to its target website's frequently changing hierarchy. This modification of a browser-based software package is performed to show that tools created to be used for personal preservation are not restricted by the structure of the target website's ephemerality.

To extrapolate the re-implementation of Archive Facebook into one that is website agnostic, I have created a synthetic social media website with a hierarchy that resembles those currently in existence but that can be manipulated to show the tool's adaptability. By doing this, tools that implement the specification inherit not only the trait of being dynamic regardless of the target website's hierarchy but also are applicable to preserving websites outside their original intention with minimal maintenance for the tools' developers.

Finally, to serve as a bridge toward better personal web archiving, I have developed a browser based preservation tool, WARCreate, that allows any webpage to be converted to the aforementioned WARC format. Providing this facility allows websites that were previously not preserved, namely, content on social media websites, to be preserved be any user that deems the content important or has a need to preserve it.

I.3 CONTRIBUTIONS

Personal web archiving is frequently performed in a sub-par fashion with tools and methods that would benefit from the advances already enjoyed in conventional web archiving. To improve the state of personal web archiving and put forth issues that need to be resolved in the creation of personal web archives, I intend on contributing the following with this thesis:

- Enumerating outstanding issues that plague personal web archiving and issues that tools built to capture content not previously preserved would face.
- Identifying and determining the commonality of hierarchy possessed by a select set of social media websites.

- Creating a means, through a remote specification, for tools that are built to capture content on social media websites to become adaptive to the sites' structure irrespective of the ephemerality of their hierarchy.
- Evaluating the specification through the modification of a currently existing tool to utilize the specification.
- Enabling users to preserve personal web content through the creation of a browser extension, WARCreate, that allows any webpage to be preserved into the WARC format, which was previously inaccessible.
- Modifying a client-side server suite to provide the facilities and capability of personal preservation tools to utilize server technologies without inappropriately exposing the data to be preserved.

I.4 THESIS ORGANIZATION

Chapter II will give information on the status quo in regards to various sorts of digital archiving. Chapter III considers issues unique to personal web archiving that currently exist or would need to be overcome when the task of preservation is handed from the crawler to the user. Chapter IV will introduce tools built for or heavily manipulated to accomplish the goals of this thesis. Chapter V will progressively build the specification to be used by personal web archiving tools to become more adaptive to hierarchical changes in social media websites. Chapter VI will utilize the specification generated in this thesis via the modification of a select set of personal web archiving tools. Chapter VI will also discuss some implementation-specific caveats that arise in creating browser-based personal web archiving tools. Chapter VII will evaluate the effectiveness of the specification for a reference implementation in adapting to changes in a synthetic social media website's change in hierarchy. Chapter VIII will discuss the conclusions drawn from using the approach of conforming to the specification as a hierarchical reference, the merits of adopting it, the contributions of this research to the field of personal web archiving and future work that could be done to extend this thesis.

CHAPTER II

BACKGROUND AND STATE OF THE ART

Understanding why the goals of this research would be useful requires one to examine the state of archiving as a whole, particularly in the non-disjoint realms of personal digital archiving, web archiving, and personal web archiving. Each of these has outstanding tasks to be resolved or considered, many of which are intractable. The importance of preserving digital content lies in that the content is largely ephemeral. Missing web pages, for example, are ubiquitous in today's browsing experience [34]. Efforts like the Firefox add-on "Synchronicity" that support the user in (re-)discovering missing webpages [33] through access to other archives and caches would be more effective if more content were preserved. Preserving more content in the ways considered to be "best practice" enables recollection of a digital resource once it is discovered and in need of retrieval. Various software endeavors and their respective services (e.g., IA's Wayback) have driven forward momentum of digital archiving but not all aspects have been translated over to personal digital archiving.

Three realms of applicability for this research are personal digital archiving, web archiving and personal web archiving (Figure 1). Issues that reside in a more specific realm might find resolution in more general realms. This is especially the case in realms that are encapsulated by another (e.g., personal web archiving within the realm of web archiving). Because of this, the discussion of these realms will get progressively less abstract.

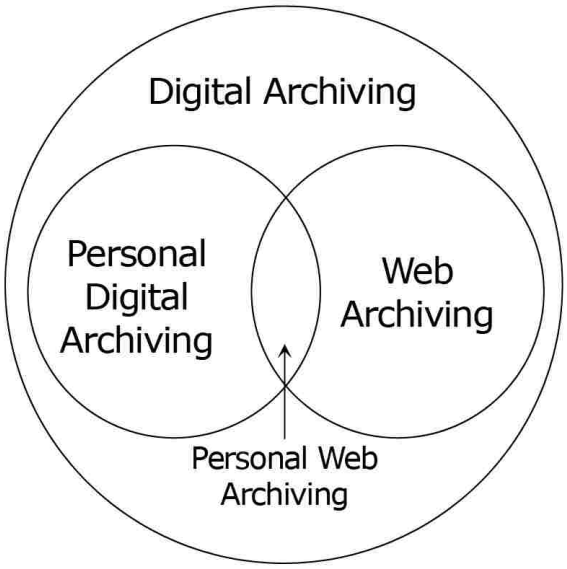


Fig. 1. To simplify the discussion of the various realms of archiving, this Euler diagram shows how each realm relates.

II.1 STATE OF PERSONAL DIGITAL ARCHIVING

Personal digital archiving spans a wide range of applications from assuring that stored content is well-backed up and easily findable to verifying that the information can be recovered if an original is ever lost. In the context of digital assets, many users are aware of good practice yet few institute a consistent backup regimen or only backup on an ad hoc basis [40]. Still then, users assume that the ethos of the LOCKSS [39] (Lots of Copies Keeps Stuff Safe) system is sound but until a resource needs to be accessed, the availability of the resource is often not verified. An individual is seldom aware of this digital brinkmanship [40] and in practice, few do much to hedge against this loss [45] in which a LOCKSS-like system would be helpful in preventing. Individuals frequently use sub-par strategies like using system backups as archives, moving files from one machine to another, moving files to another medium (e.g., CDs, floppy disks) as their archiving strategy [44, 63]. All of these are problematic and reduce the ability to recall information when needed. Sound personal archiving practices are not commonplace both because users are seldom able to implement their current strategies consistently [44] and many of their strategies give a false assurance of their soundness due to user ignorance.

As Hodge [28] discussed, with proper life cycle management, digital objects are

more likely to be preserved. Best practices in preserving digital content come down to a system of proper creation, acquisition, cataloging/identification, storage, preservation and access. For example, practices used when a digital object is created ultimately impact the ease with which the object can be digitally archived and preserved. Further, best practice requires that metadata is created as objection-creation stage and that the archiving process is made more efficient when attention is paid to issues of consistency, format, standardization and metadata description in the very beginning of the information life cycle.

II.2 STATE OF WEB ARCHIVING

Generally speaking, web archiving consists of creating archives of any content that resides on the web. The Internet Archive et al. have been major players in facilitating the preservation of content on the web and assuring metadata is attributed. The Internet Archive and Nordic National Libraries created a web crawler, Heritrix [56], to crawl websites for inclusion into the Wayback Machine. The code for the Wayback Machine is written in Java and was originally created with the intention of being open source (distinguished from here on as all lowercase wayback) so as to promote collaboration between institutions that were interested in archiving the web [56]. Prior to their efforts, no one had tried to capture a comprehensive record of the text and images contained in the documents that appeared on the web [30]. For content that has not been successfully archived by a particular organization, methods like utilizing the lexical signatures of lost web pages [32] to find content not in an archive, referencing other archiving institutions to supplement periods of time when a page was not archived [69], and referring to search engines caches that were created as a result of the indexing process [52, 54] can be used to restore and repair incompletely preserved content.

The output format of Heritrix, a WARC file, consists both of records to describe the content to be archived as well as the archived content itself. Because Heritrix experiences the web in a similar manner to a user when utilizing a web browser, in that the HTTP protocol is used to request and process remote content, Heritrix is able to capture these headers into a WARC file and use them as a basis for replay. Conversely, tools that capture only the content after it has been processed by the browser do not retain this metadata, which causes the replay experience to be incomplete and frequently inaccurate. An example WARC file is included as Appendix D,

to which line numbers given here refer. WARC files begin with a *warcinfo* record (lines 2-15), which describes the WARC file and the tool used to generate the file. After this are several WARC records that are generated through a crawl. These include the following:

- **metadata** - describes a collection of resources crawled including information such as the date of the crawl. (line 37)
- **request** - an abstraction of an HTTP request into the WARC format that allows the request to be used in the replay of the preserved content. (line 18)
- **response** - like a request header except it contains the payload of the archiving process - the webpage or resource to be archived. (line 48)

These records are concatenated together (with all binary data joined inline in a form suitable for replay) into a file representing an instance of the WARC format. Many WARC records containing multiple websites and crawls can be contained in a single file. Further, a crawl is capable of being split among multiple WARC files using a specific WARC record to guide the replay system in resolving external references needed.

Though the WARC file format is widely used, it is just a specification for a container and says nothing about the formats or semantics of the objects contained within WARC files nor about their relationships to each other [68]. While other formats exist (e.g., FOXML [22], METS [38], MPEG-21 [10], etc.) that attempt to accomplish web resource bundling, none adhere to the Open Archival Information System Reference Model (OAIS) [17]. The objective of the OAIS model is to provide a framework for the use of these bundling specifications [68]. Because of the WARC format's adoption of the model and its utilization by the most prominent web archiving organization (Internet Archive), further discussion assumes this to be the optimal format on which to base future web archiving efforts.

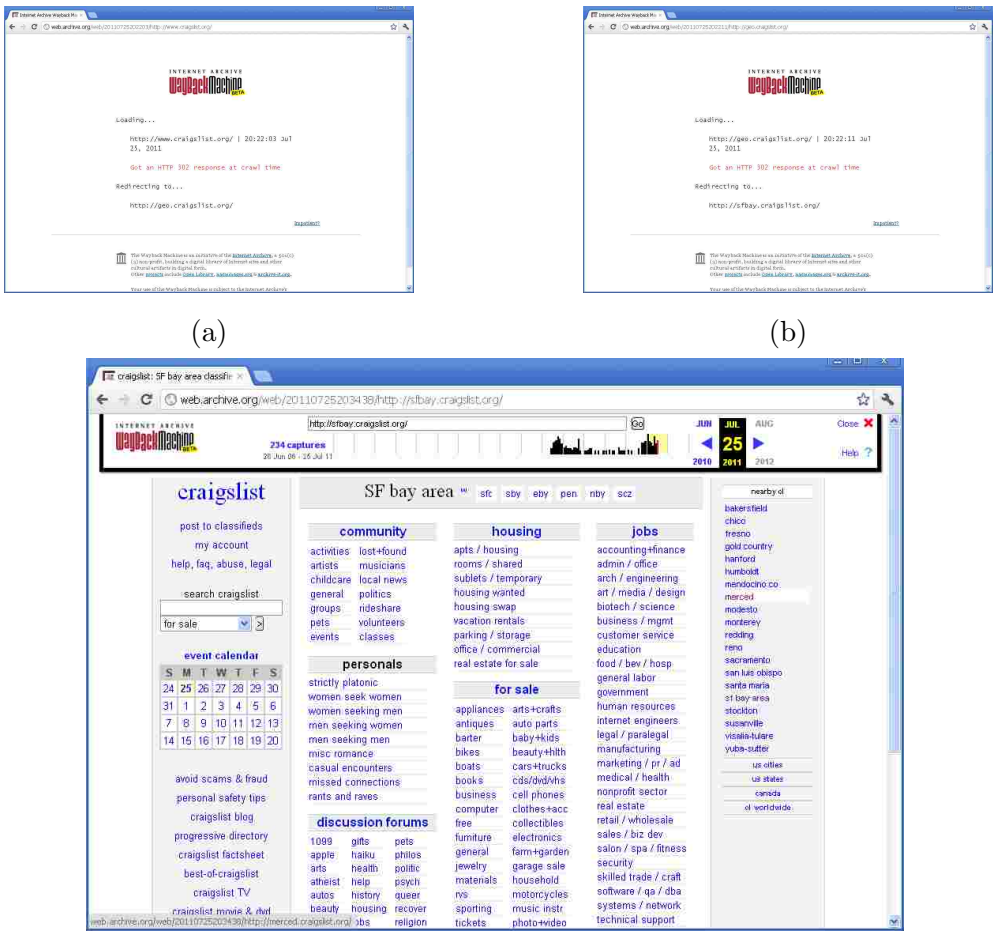


Fig. 2. Replaying the July 25, 2011 archived version of Craigslist returns the unexpected result of the crawler's original locale instead of the user's current locale.

Little has been done in the way of assuring that an archive is replayed in the manner originally intended. Archives generated from Heritrix are replayed as if being viewed by Heritrix. An example of this problem can be observed with the Internet Archive's crawler of Craigslist[1]. The archived version retained by Heritrix of Craigslist is based on the perspective of the crawer, i.e. Heritrix as run from the Internet Archive in San Francisco. A user that wished to recall the content on craiglist.org (Figure 2a) at a certain date would not have the luxury of simply entering the domain but instead is required to be familiar with the site-specific redirection scheme (Figure 2 progression from top left clockwise). This is one example where not maintaining the look and feel through archiving potentially compromises the

[1]http://craigslist.org

content targeted to be archived. The reason for this discrepancy can be seen in Figure 3, which shows the site's reliance on GeoIP. When originally generated, code that appeals to the user's perspective (in this case, the web crawler; in other cases, the user's web browser) may result in unexpected output on replay. Emulation is needed so the replay of web resources may exactly imitate legacy software [42].

```
> wget http://www.craigslist.org
--2012-08-06 18:40:09--  http://www.craigslist.org/
Resolving www.craigslist.org... 208.82.238.225
Connecting to www.craigslist.org|208.82.238.225|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://geo.craigslist.org/ [following]
--2012-08-06 18:40:09--  http://geo.craigslist.org/
Resolving geo.craigslist.org... 208.82.238.225
Connecting to geo.craigslist.org|208.82.238.225|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://norfolk.craigslist.org [following]
--2012-08-06 18:40:09--  http://norfolk.craigslist.org/
Resolving norfolk.craigslist.org... 208.82.238.225
Connecting to norfolk.craigslist.org|208.82.238.225|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
```

Fig. 3. The basis for Internet Archive's Heritrix capture of Craigslist.org can be seen with this fetch of the website using wget showing the site's reliance on GeoIP.

II.3 STATE OF PERSONAL WEB ARCHIVING

Curation of personal digital materials in online storage bears some striking similarities to the curation of similar materials stored locally [45]. The aforementioned neglect stems from a lack of current need for resource recall, the inability to sufficiently archive because of an unsound or poorly implemented processes and no standard medium to assure that the output format will be able to be read in the future. These are only a few of the numerous reasons why the practice of web archiving, and particularly personal web archiving, is in disarray. Users will often use the circular reasoning of a service supplying the backup for data they have stored on the web yet resort to poor archiving practices in assuring that the content is preserved.

For example, pictures on photo sharing websites[2] retain more metadata than those stored in a directory on a hard drive. While one usually will reference the more comprehensive collection of photos on a local machine, in the event of system failure, only then, will they attempt to recover photos from the photo sharing website. Little redundancy is put in place and where it is, the integrity of data being backed up is rarely verified until the data has been lost. Users are unwilling to put forth any curatorial effort to ensure their work is not lost [45].

The extent to which a digital object is preserved corresponds to the methods and medium used to accomplish the preservation. People archive their personal digital belongings by relying on a combination of benign neglect, sporadic backups and unsystematic file replication [45]. Tools created to alleviate the process are only as reliable as the methods and medium that the tools employ. Archive Facebook, for instance, preserves content in a directory structure navigable by a web browser. While this content is saved to disk and retained, without appropriate measures (e.g., storing of metadata, appealing to standardized preservation formats) and a standard medium (the tool suffers from the issue described in Section III.3), the content is not preserved to a degree that we would expect of an archival format like WARC.

II.4 CURRENT TOOLS

Users currently have access to a wide array of tools to accomplish the task of preserving their personal data. Some of these "tools" are simply interfaces provided by the service, offering the user a way to liberate content the service deems as belonging to the user into a replayable format. Other tools suffer from varying degrees of sub-optimality in terms of the output they offer the users. Here I take a look at each tool's output in comparison to all others. A high-level comparison of these tools to one another is available in Appendix B.

II.4.1 FACEBOOK DATA DUMP

Facebook allows one to "download a copy" of their Facebook data. To accomplishing this, one must access General Account Settings[3], select "Download a copy", enter the password, and submit the request. Facebook then gathers the information that it feels is owned by the user, bundles it up in a navigable set of web pages devoid

[2] e.g., http://www.flickr.com

[3] https://www.facebook.com/settings

of the Facebook styling, and e-mails the user with a link to access the bundled files. An example of the output can be seen in Figure 4.

The process of the user being notified varies to a degree undocumented by Facebook but likely has a correlation with the number of resources attributed to the user along with the amount of processing needed to decide ownership of the content. Empirical tests on different user accounts do not show a direct correlation of time before the user is notified and frequently resulted in a response with a large delay (as much as a full day) to no response at all. The end-result could loosely be considered an archive but lacks integrity, in that information was removed from the context of Facebook and replaying the "archive" would not result in an experience similar to the replaying an archive from wayback.

Fig. 4. Facebook's "download a copy of your data" feature results in a navigable set of locally accessible webpages with a selection of resources that Facebook determined as appropriate for an "archive" and that also belonged to the user. The interface deviates greatly from its original context. This selective exclusion of content as well as the deviation from the original context produces an "archive" of questionable integrity.

Noticeably absent from the data dump is content in which the user might have been associated but does not "belong" to the user, as determined by the dump procedure. Beyond simply maintaining the look-and-feel, intentionally excluding content from a preserved medium sets a bad premise for preserving content qua archives. If the purpose is to preserve, as is the user's intention with the use of this tool, and data related to the user that would be expected to be included is instead

excluded, the task of preservation was not performed to an acceptable degree.

II.4.2 GOOGLE TAKEOUT

Google Takeout[4] is a service provided by Google to allow users of Google's services (including their social media counterpart, Google+) to download all data that one has generated. Through an AJAX-driven web interface, Takeout allows a user to select from "+1s", "Buzz", "Circles", "Contact", "Docs", "Picasa Web Albums", "Profile", "Stream" and "Voice" data to be included in an "archive". After collecting the information, Takeout provides a link to download the .zip file. This file contains an extremely comprehensive set of data representative of the content of one's account organized into a directory structure, much like Archive Facebook. Unlike Archive Facebook, however, the data is not stenciled in the original website design of the contents' respective origin. Also, unlike both Archive Facebook and the Facebook data dump, there is no way to navigate the archive and view the resources' contents in a single medium (e.g., the web browser). Both of these differences are appropriate, as Google appears to supply data that might not be readable in a web browser (e.g., vCard files for the "Contacts" portion of the Takeout output).

II.4.3 SEQUENTIAL "SAVE WEBPAGE AS"

Most browsers provide the facility to save the webpage currently being viewed to disk. The output of this operation is similar to that of Archive Facebook's. Archive Facebook has the advantage of allowing pages to be pre-processed (e.g., continuously scrolling to the bottom of a page until no more new content loads) yet "save webpage as" is not tied to a particular website, so it is more general purpose. If a similar set of webpages were sequentially (as in Section VI.2.1) and manually saved using this primitive scheme of backup (for which "save webpage as" also suffers per Section III.3), the produced backups are not directly accessible from one another using backed up content. With Archive Facebook, the backup is fully navigable. With the Wayback Machine, archives of a website are fully navigable. The general purpose nature of this procedure, while universally useful, does not produce content in an archival format, does not exhibit pre-processing of webpages without manual intervention, and does not allow a series of backups from a single webpage to be

[4]http://www.google.com/takeout

inter-navigable, so it not nearly as suitable for archiving as other tools listed.

A post-processing procedure of defining how the backed up pages align with one another along with converting the content archived into a recognized archival form like WARC would be a step for the better in this procedure. Unfortunately, metadata about the original context of the backup as well as any headers needed to replay the content in its original form are absent, so improvement on this procedure would be fruitless without modifications to its foundation.

II.4.4 OPENSOCIAL

OpenSocial[5] is a bridge-like API that serves as a single medium of interfacing with a variety of social media websites. The service works by having a user log on to the target website, access a page on the target website's domain representative of the conduit for external data access, agree to let external services use this data, then build applications based on a received authenticated key for access to the target's website. There are a few issues with this approach that are addressed by appealing to the specification proposed in this thesis.

Firstly, OpenSocial is opt-in. Notably absent from the list of available social media networks is Facebook. By relying on the service to green light the external access process, a barrier is put in place in preserving the content for the user on the social media website. Further, if the target website is supported, the data received from the target website suffers from similar issues as referenced in Section I.1 wherein a target website decides what information about a user to liberate. An additional problem with programming against this sort of meta API is that the results are likely similar to that of a Facebook dump at best, i.e. it is unlikely that the service (Facebook in this case) would provide additional data or design not already included in the Facebook data dump. Because OpenSocial requires a service to opt-in, can be potentially limited in the data it receives from the target website, and does not preserve archive integrity through output with the target site's look-and-feel, OpenSocial is unsuitable for personal web archiving in the degree paralleled by the use case tools.

II.4.5 WARC-TOOLS

The Hanzo Archives WARC Tools suite[6] is a set of core libraries/APIs and

[5] http://code.google.com/apis/opensocial/

[6] http://www.hanzoarchives.com/solutions/open_source/projects

command-line tools for full-text and metadata-based search of archives in WARC format [68]. Its implementation is completely decoupled from the Internet Archive's open source wayback offerings, which allows the programs in the package to function without the technical overhead of a user installing a personal instance of wayback. Because of this decoupling, the tools suffer from an issue similar to WARCreate in that their implementation of the full WARC ISO standard is incomplete, causing failures in validation where input should pass. However, WARCreate relies on the external cdx-indexer for validation of generated WARC files (as documented in Section VI.3.1) and, thus, is able to generate content that will work in the replay system. Though the Python programs in the warc-tools package do not generate original content (rather, they serve as the foundation for the package described in Section II.4.7), observing their failure with simple validation of officially generated (i.e., generated by IA's Heritrix instance) WARC files sets a poor premise for packages to use it as a basis.

II.4.6 WGET WITH AUTHENTICATION

GNU Wget[7] is a free software package for retrieving files using HTTP, HTTPS and FTP. Wget is frequently used to fetch and store content from the web en-massse. Because of its command-line accessible interface, many use the tool in conjunction with scripts as a first step in retrieving the desired data. Wget allows parameter specification using command-line flags to supply authentication credentials as well as the ability to use an external file as a source for cookies. However, the perspective issue akin to delegating the archiving task to Heritrix (Section II.2) is present in an approach using wget or any tool that causes the replay experience to deviate from the original form. Further, wget does not contain support for Javascript[8], a requirement for most AJAX-heavy social media websites, so it is unsuitable in its base form for retrieving and archiving content on social media websites to a satisfactory degree.

[7] http://www.gnu.org/software/wget/

[8] http://wget.addictivecode.org/FeatureSpecifications/JavaScript

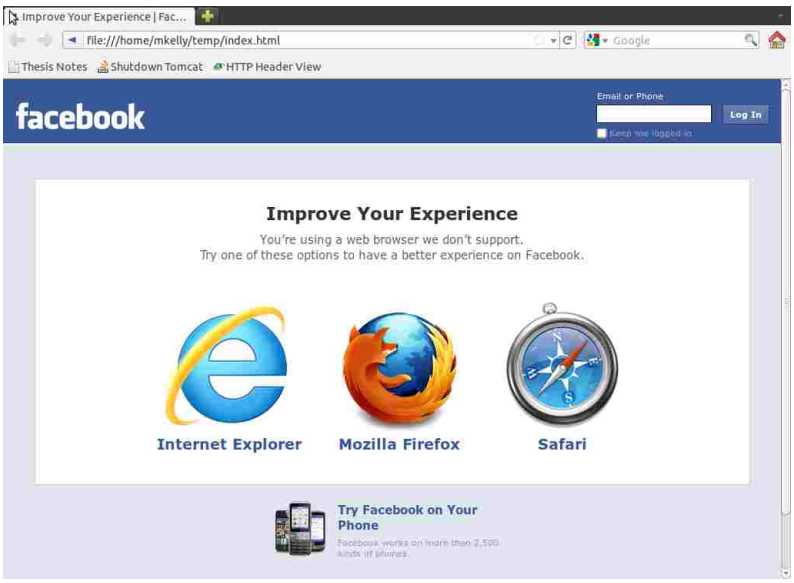


Fig. 5. Social media websites expect users to experience them with a conventional web browser and not a fetching tool. That this is enforced to the top level of the website when accessing it with wget is a red flag that content, were it to be archived, would likely not have its look-and-feel preserved and would very definitely be incomplete because of the lack of Javascript support by fetching tools.

Prior to manually entering the credentials needed to authenticate with a social media service (e.g., Facebook, as shown in Figure 5), issues start to arise. To even expect an acceptable replay experience, the fetched page must be viewed in a web browser rather than a fetching tool. Services will often exclude tools and browsers they do not support (further emphasizing the issue of preserving look-and-feel and perspective as documented in Section II.2), which makes mediums used specifically for archiving sub-optimal. With the preference of social media websites to have their users experience their website in a browser context and wget's lack of support for Javascript, even with authentication it is insufficient for personal web archiving.

II.4.7 WGET-WARC

wget-warc[9] is a patch onto "wget" that allows the program to output its data to WARC files. With some work, one can specify wget to retain response headers but there is no way for it to retain request headers [6]. The program also provides a clean way to store redirects and 404 responses. wget-warc utilizes the warc-tools

[9]https://github.com/alard/wget-warc

package from Hanzo Archives, providing the missing data fetch element that Hanzo's offering does not provide. Because wget-warc utilizes wget, however, using it as a tool to archive social media website will lead to the same problems as utilizing wget (Figure 5). The project has great potential in retaining the headers, as required by the WARC format but is not very user friendly (its compilation failed on two different Linux distributions, even after some work). The tool will be much more useful once it is integrated into a wrapper to allow it to process browser-based technologies (e.g., Javascript), thus making it more relevant to personal web archiving of social media websites.

II.5 SUMMARY

Chapter II considers the state of various relevant forms of archiving and how being cognizant of the fields' current offerings and needs help to provide justification for the objectives of this thesis. Sections II.1, II.2, and II.3 describe each realm, how they relate to each other and how aspects of some are still not adapted to the others. Section II.4 considered facilities provided by social media services and open-source tools that attempt to enable users to preserve their information. Unfortunately, none of these approaches are optimal for personal web archiving.

CHAPTER III

CONCERNS UNIQUE TO PERSONAL WEB ARCHIVING BEHIND AUTHENTICATION

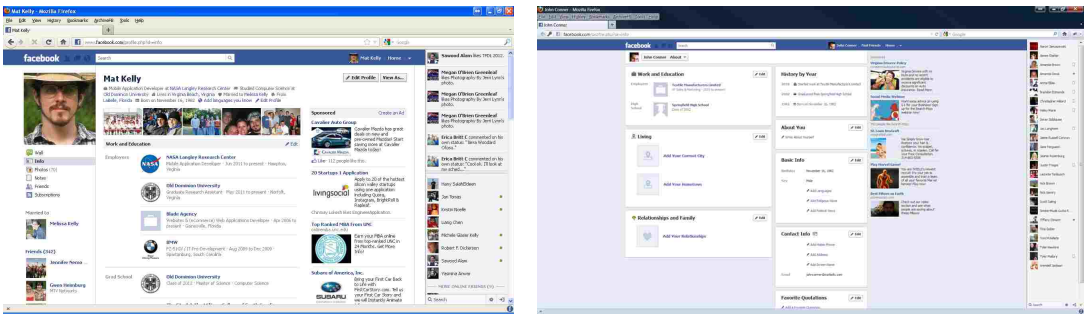
Personal web archiving exhibits some features not present in conventional web archiving that should be addressed. Considering these prior to moving forward with new methods will allow the methods to have a foundation in both accomplishing their task while taking into account some of the caveats that are likely to come about. These caveats arise when the task of personal web archiving is addressed in a naive manner similar to conventional web archiving. Though there are likely to be further concerns as the processes of archiving this content matures in the future, the issues of naïve URI cardinality, authentication, privacy, and security in regards to personal web archiving will be addressed. Topics relevant to web archiving in general that will be discussed are of archive integrity, archiving vs. backing up and the inconsistency in methods of obtaining data that plague web archiving as a whole.

III.1 NAÏVE URI CARDINALITY

Once a user is authenticated with a target website, the contents of a single URI will likely vary from user to user. In the sample case of Facebook, a user's news feed is composed of the contents of the recent updates from a respective user's friends. Being that it is unlikely that two users have an identical set of friends (and even if they do), the contents of two users' news feeds will be different while still being accessed with the same URI (e.g., www.facebook.com). Much of the content on the web is assumed to contain the same data when accessed by two different users. Even content within the Deep Web [8] with the same URI and explicit (GET request) parameters will likely result in identical content, especially if the process of accessing the content has no side effects. As content on the web becomes more dynamically generated [36,58], hidden behind web forms and other kinds of query interfaces [71], it becomes less accessible to crawlers[1]. For example, if two users access a URI that is not

[1]Though content on social media websites fits Raghavan's definition of "dynamic" [58] and thus is the target for the crawler, it is not, "form generated" and so contradictorily does not fit the criteria.

indexed by search engines (through robots.txt [2] enforcement or simply because it is not linked from elsewhere) residing at http://www.example.com/?key=secretKey, the content could be different because the state of the resource between accesses is not guaranteed (i.e., the user can never step into the same river twice [61, 62]). One reason for this is that pages include code that executes on the client machine to retrieve further tailored content [58] (frequently implemented as Ajax calls). If this content requires authentication, it is still possible that the content will remain the same between different users, but it is the nature of social websites to tailor content to its users. Even if the content when accessing http://www.example.com/?key=secretKey while authenticated is nearly identical, any tailoring to the user (even a content variation as subtle as a "Hello Username" message) will result in unique content. One cannot assume that accessing a URI will result in the same content or even the same design when accessed by different users. This is especially true in the case of web archives when the representation of the resource (e.g., the HTML) is retained but parameters originally sent to request the resource's representation (e.g., cookie information, credentials) are not sent to the replay system. Figure 6 shows two different Facebook users' returned content after having accessed the same URI (http://www.facebook.com/profile.php?sk=info). Normally, only the content would be tailored to the user but because of the website's personalization, the page is completely different though temporally equivalent. On the surface web, only a temporal difference would result in this drastic of a difference.



(a) Standard Facebook interface (b) Facebook timeline interface

Fig. 6. URIs can not be used to guarantee what content is returned when different users access the URI because of site personalization. The tailoring of preferences here shows a user that is retaining the look-and-feel of the previous version of Facebook (6a) and the interface presented to a user that has opted into the Facebook Timeline interface (6b). Though two different users are accessing content using the same URI, the resulting content is drastically different because of the user-based content tailoring.

Less subtle differences in the content displayed to different users when accessing the same URI are becoming more commonplace as websites enable users to control privacy settings. By using these privacy settings, a user is able to restrict or allow specified content to be displayed with a scope that can span any of "publicly accessible", "only friends", "only me" or any ad hoc subset. Though two users might be friends (in the context of social media websites) with a third, the common friend can potentially tailor what each user sees on a per user basis. This implies that, for the most part, the content displayed to a user from a system that allows tailoring, is almost always guaranteed to be unique.

As the content can greatly vary between users and the same URI when accessed by different users can result in different content; in order to archive the content from the perspective of a user, some unobtrusive identifier must be added to the means of accessing the archive to assure that the content from the desired perspective is served. Such an identifier will assure that when two users access the same URI, a secondary identifier will provide the facility to serve the user the appropriate concept.

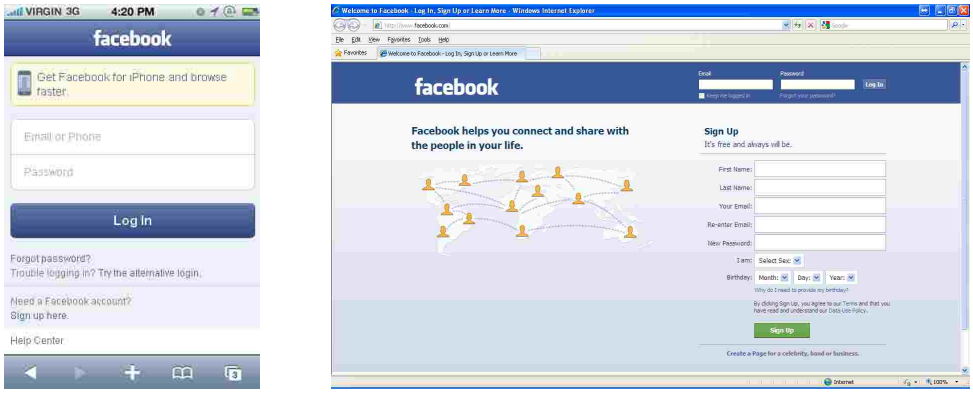
One way to accomplish this requirement of assuring consistent access to a unique archive is to add "perspective" data to the URI in the location where the username would normally reside [9]. An example for a user with a unique identifier of

12345 would embed this data with a result like http://12345@socialmediawebsite.com/path/to/further/resources.php?p1=foo. The overloading of this attribute remains semantic, as the "username" field in the URI scheme is still representative of a unique identifier but expanded to be a signifier to other tools to handle URIs with this addition differently. In the case of HTTP, this attribute is rarely included in the user scheme and different means of authentication are normally used (e.g., requiring credentials on access of the URI that does not contain the username:password@urischeme form). An implementation-based issue with this approach is that in the default implementation (at archive.org), Wayback Machine's crawler (Heritrix) uses a filter [3] to strip away certain information including the username field preceding the hostname. Because of this, a different approach must be used to accomplish perspective designation of an archive. This is discussed further in Section III.2. As is evidenced here, Heritrix's default functionality is too generic to be applied to personal web archiving of content requiring authentication. The lack of context and stripping of an external means to represent this context make it unwieldy for a casual user. A user who would like to archive certain content on a specific website would need to explicitly define URIs to crawl.

III.2 CONTEXT

When viewing a webpage from various perspectives, be it one of the variety of web browsers available or from different devices (e.g., mobile phone, PC), it is possible that different content is displayed based on the user's choice of device (Figure 7). Some websites serve a completely separate, often optimized version of a website to those on mobile devices that might be restricted by bandwidth, screen real estate or any number of limitations that a mobile device imposes. The code and markup behind a webpage might also be tailored to serve certain information only appropriate to users of a certain browser, potentially even limiting which browser may view a website, as was common in the browser wars [70]. This behavior is still found on systems where reliability of experience is crucial and the website administrators have taken the route of excluding rather than being accessible. For these and other reasons, it is easy to imagine two webpages (even those on the surface web behind no authentication) being rendered differently when viewed by two people or by a single person on multiple devices. As personal archives are much more susceptible to this tailoring because of the desire of social media websites' users for ubiquitous access

to a service, it is important to address these deviations in user experience and how they relate to personal web archiving.



(a) Mobile Facebook display (b) Standard Facebook display

Fig. 7. When accessing facebook.com from a mobile device (7a), the content supplied to the user is tailored to the user's available screen width. Where the screen width is less predictable but often wider, as is the case with a PC running Internet Explorer (7b), the user is supplied content with much more detail.

Information about a user's perspective when visiting a website is identified by a user agent string, representative of identifying information (e.g., choice of browser, current platform) and a few other pieces of information, essentially a digital fingerprint [20], to allow the sniffing of a user's browsing attributes [26]. As was more common in the past, when a user was prevented from accessing a website because of one of these attributes (e.g., blocking any users that are not on a Macintosh from a Macintosh Fan Club website or the exclusion imposed by Facebook in Section II.4.6), this user agent information could be spoofed (or falsified) to circumvent the restriction. Spoofing, as in Figure 9, also has the constructive use in testing to assure no restrictions of this sort are being accidentally imposed by the webmaster. Even with spoofing, how the page appears from the spoofed perspective cannot be accurately observed without the further assistance of a corresponding rendering engine.

```
<!--[if lt IE 5]>
Your browser is too old and cannot render this content.
<![endif]   ->
<!--[if gte IE 9]>
...features not supported by version of IE prior to 9...
<![endif]-->
```

Fig. 8. Internet Explorer provides a way of exploiting the constructs of HTML comments to provide code that is only pertinent to a subset of versions of the browser.

An example of constructively using browser spoofing would be in testing code or markup that is tailored to a specific version of Microsoft's web browser, Internet Explorer (IE). Since version five[2] IE has allowed unobtrusive HTML comments of a certain form to be rendered only by those that satisfy the condition. An example of this is shown in Figure 8. This content, enclosed in an HTML comment tag from the perspective of any browser but IE, will render differently between users that are using IE version 4 and version 9 and will show no content generated from within the comment for users of browsers other than IE. In this instance, IE versions less than version 9 are unable to natively render the content within the second conditional and so are appropriately not shown this content so as to not confuse users with non-functional content. This filtering is performed client-side and could be overcome with user agent spoofing. In addition to content exclusion, a user may be served what the webmaster believes is a more appropriate display of the content, potentially leveraging features that are only available and are appropriate on a certain platform. An example of this would be forwarding a user to a mobile version of a website that utilizes the GPS functionality in a smart phone that would not be appropriate on a user accessing the same website from a PC without this functionality. Content might also be dynamically generated based a user's choice of browser, potentially utilizing a reliable browser detection library (e.g., QuirksMode's BrowserDetect[3]) to serve only appropriate media that the webmaster believes will be compatible and optimally experienced by the user. The webmaster could also potentially exclude access to content regardless of user agent spoofing, as the above library does not rely on the value being spoofed via the user-agent request header

[2]http://msdn.microsoft.com/en-us/library/ms537512.aspx
[3]http://www.quirksmode.org/js/detect.html

to reliably detect the user's browser. Instead, the script checks for support for various client-side attributes and operations that are present in a browser, secondarily relies on the "navigator.vendor" attribute for identification and finally falls back on the "navigator.userAgent" attribute if the other methods of identification fail. Performing browser detection in this way circumvents many of the methods tools use to accomplish spoofing but also provides a way for developers to reliably serve content only meant for a specific subset of their users.

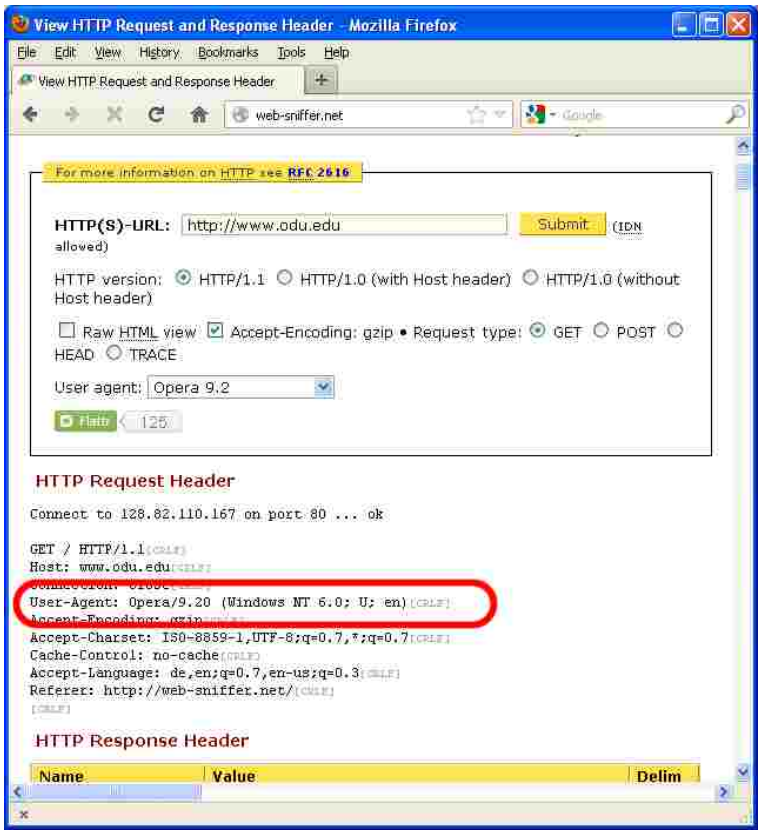


Fig. 9. Websites like web-sniffer.net allow a user to spoof their user-agent to determine if different results are produced when various browsers are visited. Browser-based plugin approaches also exist but by using web-sniffer, a user is able to see the method used (modification of HTTP headers) to accomplish the spoofing. Note the spoofing of the Opera web browser while Mozilla Firefox is being used.

If a user wished to archive two versions of a website from two different perspectives (e.g., browser, platform) and view them from either in the future, the content might not be displayed properly yet that might not matter. The device from which a user

views his archive should not limit whether the archive is viewable, but an attempt should be made to display the archive that contains the most similarity to the user's current perspective. Retaining this information to be included with the archive should be as simple as capturing this metadata at the time of archiving, but the Wayback Machine does not natively support user agent switching and instead only serves the representation it received at crawl time.

Because Wayback strips portions of the URI it considered superfluous to archiving the page (Section III.1), encoding user agent information into the URI would be problematic. Instead, a more reliable way to preserve this information would be to encode it as metadata in the WARC records. Metadata is information that enables and documents the long-term preservation and access to digital objects [68]. Retaining this information at the creation stage of the digital objects is preferential for good practice, as described in Section II.1. Because the implementation-agnostic case (i.e. appealing to the WARC format independent of wayback and Heritrix) is useful to explore for further developing the applications that utilize the WARC format, encoding user agent information in the URI will be discussed further here. Doing so will reinforce the lower degree of qualitative optimality of considering an encoding-based scheme over one that appeals more to the format itself.

With encoding the user agent into the URI, we wish to retain semantic, simple URI schemes of the resulting archive and generalize the scheme to be applicable to a variety of social media websites. Encoding too much information within the URI scheme might be counterproductive to these goals. To consider a possible encoding scheme while leveraging the perspective specification described above and ignoring the implementation-specific filtering by Heritrix, this information can be encoded in what would normally be used as the password field in a URI. With the Facebook example: http://perspective:useragentinfo...@facebook.com, it is unclear how to encode this information to be comprehensive of all of the information in the user-agent string and still be succinct. Extracting all of the content and appending them in a fashion akin to appending variable values to the end of a URI string (e.g., http://www.example.com/index.php?user=john&color=blue) is not conducive to our goals, as even a scheme with limited browser attributes would require URIs like "http://myusername:engine=Mozilla&plaform=AppleWebKit&plaformversion=7B405&...".

Unlike overloading the username portion of the URI scheme, this sort of information loading is an abuse of the original intention of the password attribute and would not scale well when integrating the archives with other systems that use the field for its intended purpose. Because of these issues, it would be difficult to represent all possible permutations of perspective without a reference to external encoding and that would still then contain a combinatorial number of variants.

Two non-mutually exclusive alternatives to this scheme would be to either include the user agent information as WARC metadata (suggested above) and/or inject the information into the target page for later reference. The first case is optimal, as it is the more semantic option and the WARC format is designed to be extended in this way. Unfortunately, from the client side perspective and the current offering of the Wayback Machine and the open source wayback[4] package, this user agent information is not accessible to the end-user, which makes storing it in a WARC metadata record useless without a customized wayback build modified to expose this information. An additional offering from the Internet Archive's Archive-It[5] website has implemented a way to attach additional metadata to an archive by using an external database, but because of the overhead, this would be impractical for a casual user.

The second alternative is more obtrusive on the target document and less semantic than the aforementioned ideal approach but is very accessible to the end-user, who is the target of this study. At time of archiving, the user agent can be collected and injected into the target HTML page as HTML metadata, rather than WARC metadata. This information is directly accessible on the client-side via Javascript. Because of this obtrusion, maintaining archive integrity is quite important. Because this injection technique is the most accessible of the techniques discussed, considering it as a use case for maintaining archive integrity will also be discussed further in Section III.4.3.

III.3 ARCHIVING VERSUS BACKING UP

As documented in Section II.1, users are often confused as to what constitutes an archive over a backup, or they perform their backups using sub-par methods that make recall of resources difficult and assurance of the resources' existence difficult

[4]https://github.com/internetarchive/wayback
[5]http://archive-it.org

to verify. While institutional archiving efforts are making great strides forward, consumers are unintentionally flirting with digital brinkmanship in regards to method of archiving [40]. It is not unusual for consumers to write the most valuable of their files to external media [40], which are prone to decay and is unwieldy when the media turns into a stack of media with little-to-no metadata and immense overhead for one to recall a desired resource. Some even move files from one machine to another, presuming the data is stored and safe in a folder with a label like, "My Old Documents" yet moving files from one PC to its successor is not actually creating an archive [40]. To emphasize that which is lacking from these sort of backups to sufficiently constitute them as an archive over a backup, this sort of folder movement can be compared to the WARC format, which contains the headers necessary for replay. In addition to also preserving metadata about the process executed to preserve the data and additional information about the data itself, creating a backup using the WARC format allows for resources to be wholly portable, i.e., many resources contained in a single file. For preserved content to be accessible to the resources contained within the backup, the references (e.g., the URIs or file paths) need to be manipulated prior to preservation. With conventions like WARC, this rewriting is done at runtime of the replay system, allowing the original content to be maintained in the archive.

Many are moving away from physical storage and relying on social media, free unlimited storage e-mail space and other online services (particularly in social media) to be a redundant means of backup, again finding appeal in LOCKSS. Individuals use these services as a safety net for rescuing their digital belongings [40] where they once referred to a stack of discs. These services often format the information in a way fitting to the service (a 300 pixels per inch (ppi) image might be scaled down to 72ppi to reduce file size), which leads to a new problem of deciphering, "which copy is the best" and, "what are my options". The difference here should be clear. While the LOCKSS ethos assures that some backup is retained (assuming users occasionally assure that the content still exists, as previously discussed), without metadata, these copies still exhibit the same problems as backups, which archiving would prevent in attributing metadata. A further requirement of a backup being an archive is data portability. Utilizing the WARC format is a means to assure this.

III.4 MAINTAINING PRIVACY WITHOUT AUTHENTICATION

Some people feel that everything on the web is in the public domain [43] though the means one uses to obtaining this data is controversial [55]. With the approach of obstructing this public access to information on the web via the walled garden of authentication, social media websites are attempting to assure that only the information that a user of the service wants exposed will be exposed. With this assumption, users rely on the service to protect the data, but this protection scheme is the root cause of the difficulty in archiving the information when the user wants to liberate it for the purposes of service-independent archiving.

Retaining privacy without authentication for this data and still making it quick and easy to retrieve and replay requires a scheme of protection. Various approaches can be considered with two dimensions: degrees of encryption and centralization. The sweet spot for these two is discussed here.

Transparency of implementation is debatable, as often the level of protection a security scheme provides is inversely proportional to the amount of information that is known about the scheme being used. Ideally, advances in encryption could be applied here but many of these schemes are expensive, require a central server and are impractical from the context of a browser. As we hope to overcome the barrier of authentication by leveraging the browser, we will consider approaches toward security and privacy that can take advantage of the context of the web browser by the user and emphasize decentralization. By emphasizing decentralization, archives will retain a greater degree of portability and the process of implementing the specification will remain more accessible without the undue hindrance of an external service. The goal is to explore the optimal degree of security while still making it easy to obtain and tailor the level of security that a user desires.

The initial approach explored is that of symmetric-key-based cryptography. Through providing this simple means of encryption, a user will be able to ensure some (albeit small) degree of protection is used. We can leverage this scheme even further by providing a relevant key to the data prior to encryption with a user-specified key. This key would consist of a unique identifier representative of the user on the network, e.g., a user ID. Other secure alternatives [4] are currently being developed to accomplish the goals of generating asymmetric key pairs and are built-in the browser, i.e. they do not require by an external library. The merits and pitfalls of using this symmetric key based approach are discussed below.

The difference between encoding and encryption should be clarified, as both are

utilized by the tools built and manipulated in this thesis. Encoding consists of transforming data with the intention of usability and is utilized so that it can be consumed or transported to a target system. An example of this is image data, which is frequently transferred as an ASCII string representing the encoded form (usually base64) of the data needed for the image to be reconstructed. Encryption, however, has the intention of keeping data secret while utilizing encoding and some other security measure.

For the use cases (Chapter VI) described in this paper, a symmetric key-based approach is used. This simple approach usually relies on a shared key by multiple parties. This key is used for the initial encryption and the eventual decryption of the data. In the case of personal archives, both parties are frequently the same user. Transmitting this key is frequently the downfall of symmetric key approaches, as a man-in-the-middle attack can be used to intercept the data and brute force the key to expose the data. The approach performed by WARCreate never transmits this key but only transmits the data, so does it not suffer from this issue. Data is encrypted with the key prior to transmission. When the encrypted data is to be retrieved, it is largely nonsensical if the user does not know the key. The key is entered by the user on the client side once the gibberish is received (or supplied to the tool beforehand and retained) and used as the symmetric key for decryption, just as if the data were being sent to another user. The implementation details of accomplishing symmetric key based encryption are described in Section III.4.1.

III.4.1 OVERHEAD ANALYSIS AND WHAT IS LOST BY USING ENCRYPTION

With conventional key-based systems, the intention is to supply the key (e.g., a user's password) once, process that data and make it difficult to reverse the processing and obtain the original data. Verification that this data is correct is a matter of taking new input, running the same processing procedure and verifying that the results match. Using an RSA-like [1] method via a public and private key pair requires the overhead of a remote server-based solution to be effective. A weakness in stored key approaches lies in the exposure of the key to the attacker. Rather than relying on a stored key approach, a simpler method that also allows the data to be decrypted while not requiring the overhead of a remote server is preferable.

The more suitable approach is to use a symmetric key. Using symmetric keys

allows the content, represented in a WARC file as ciphertext, to be decoded only if the key is known by the user. This key is best implemented by way of using a standard, consistent hashing function on the concatenation of the key and the data. An extension to this would be to include the hashed key at a location in the ciphertext based on the length of the original key. A thorough analysis, which is beyond the scope of this research, would have to be done to assure that this method is reasonably secure against rainbow table attacks[6] at deciphering the content.

No browser-based decentralized approach will be as strong as a server-based solution. A solely client-side approach would use Javascript. Javascript suffers from issues of runtime malleability, shortcomings in system primitives needed for true cryptography and a variety of problems [49] that make it less than ideal for cryptographic implementations. However, adopting a server-based solution too tightly couples a potential personal web archiving tool to a single point of failure. The overhead required to encrypt the data is *O(n)* if a simple combination of base64 encoding and RC4 encryption is used. Such implementations are natively available in Javascript[7] and sufficiently secure for further discussion on less implementation specific issues.

III.4.2 FURTHER DISCUSSION ON CENTRALIZED VERSUS DECENTRALIZED APPROACHES

Aside from the case of enforcing privacy without authentication, both a partially/-fully centralized and a completely decentralized approach at personal web archiving have advantages and disadvantages. The ability to utilize external resources weighs in favor of having some element of centralization or some form of external server access to accomplish the process. This comes at the expense of loss in privacy and increases the potential to malfunction as tools on external systems need to communicate. On the other side of the spectrum, a completely decentralized approach is too extreme to be applicable to web archiving, where the resources are almost always located on a remote machine. A server-based approach could use protocols like OAuth [27, 60] or OpenID[8] to ensure that only the user that created the archive can

[6] Simpler encryption schemes, like those using symmetric keys, are especially susceptible to rainbow table attacks. The gist of the attack employs using pre-computed hashes to match up with encrypted keys.

[7] http://code.google.com/p/crypto-js/

[8] http://openid.net/

subsequently access it. If these services were to get compromised, go down, or suffer from data loss of authentication credentials, personal archives created might become inaccessible. Relying on external services for the core function of accessing an archive increases the potential for these problematic scenarios. The specification proposed in Chapter V is a guide for the tools. If it were to become outdated, removed from the web or otherwise become inaccessible, the cached version of the specification would remain suitable for as long as the previously existing specification were applicable (e.g., until it would normally be updated). It is recommended that tools based on the specification cache the respective (to the target service) implementation of the specification for an incident such as the one described. Unlike a tool relying on the external service of the specification, the issues with tools relying on authentication services (if implemented correctly) would not be overcome with caching due to the inherent reliance on such authentication systems on maintaining the allowance of external access.

III.4.3 ARCHIVE INTEGRITY

Preservation and archival of the digital born media is not trivial and can contain data quality issues [66]. Two considerations should be addressed in regard to archive integrity, one abstract and one implementation-specific. When data is collected by a crawler, it is normally not transformed in any way, as a conventional web crawler like Googlebot only indexes metadata and sometimes a cached copy that is hardly sufficient to be considered an archive (Section III.3), though efforts have been made in using these caches as the basis for archival construction after-the-fact [50]. From an end-user perspective, Heritrix appears to tailor crawled and archived pages to be replayed in the Wayback Machine by appending additional archive metadata and graphical user elements as in Figure 10. However, as documented in Wayback's administrator manual [5], the content originally archived can be viewed by the end-user by adjusting the parameters queried to the Wayback Machine[9]. This behavior also exists in the open source wayback. Often because archived content no longer exists in its original form, there is not a way to verify that content archived at a particular time, even by Heritrix, is identical to the representation in a WARC

[9]To accomplish this, the URI needs to be manipulated to add the string "id_" to the end of the timestamp of the archive being viewed, e.g., http://web.archive.org/web/20110311013223/http://www.google.com/ to http://web.archive.org/web/20110311013223id_/http://www.google.com/.

file because the source of the archive could have changed. Assuming tools created based on the standard deem it necessary to modify the content in lieu of preserving the content's original form (e.g., for usability), a list of modifications that have been performed on the content's original form (a digital paper trail [65]) should be presented before storing the data in a WARC file.

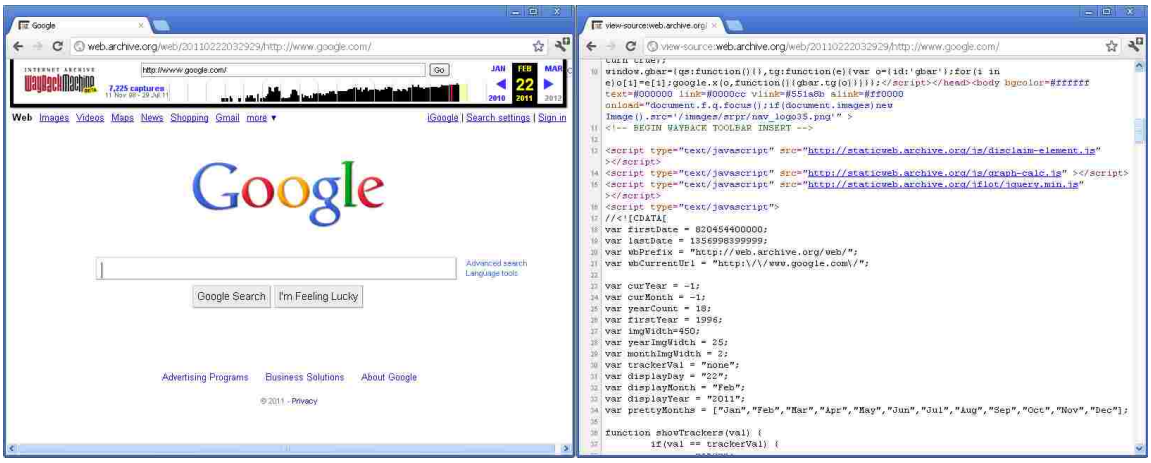


Fig. 10. Upon replay, it would appear (left) that the archive has been decorated with user interface elements by the Internet Archive to allow users to navigate between temporally different versions of the same archived page. The source code (right) seems to confirm this with the addition of various scripting that compromises the integrity of the archive so that a user cannot be sure they are experiencing the content in its original form; however, the content in its original form does not resolve URIs in a way that makes it usable on replay, so this URI rewriting procedure is necessary for a suitable end-user experience.

Providing a means to represent how content has been modified only increases the likelihood, and at worst does not affect the likelihood (i.e. it is not hindered), that the content archived is consistent with its original form. If it is necessary to first transform content in any way prior to preservation, as is done in converting images to their binary representation, this should be documented in the WARC file and attributed to the archived content[10]. An example use case where documenting the changes made to the webpages that were archived can be supposed in Archive Facebook. In the add-on, URIs on a webpage are transformed from their absolute

[10]The WARC format is inherently extensible, so a representation as simple as the output of a "diff" tool would ensure that the original representation could be restored if manipulation of the archive was necessary.

references (e.g., http://www.facebook.com/resource.html) to a reference relative to other pages (e.g., ./resource.html) that are archived in the same session to assure that the "replay" of the backed up content is navigable by the end-user. Because Archive Facebook does not currently utilize the WARC format, the application of documenting changes for the sake of archive integrity is largely moot. If Archive Facebook were to utilize the WARC format, rewriting URIs to allow navigation between pages archives would be unnecessary, as wayback prepends all URIs with the hostname on which the wayback instance resides, making references to pages that should be accessible from one page to another a process handled by the replay system, usually wayback itself.

III.5 SUMMARY

This chapter considered issues that exist in personal web archiving and are especially of concern to those that wish to accomplish it by way of a web browser. The naïve URI cardinality issue of Section III.1 addressed the matter that URI alone is insufficient for serving as a reference to an archived resource, as the same URI can represent an infinite number of variations of content at a location. Section III.2 highlighted a large problem that occurs with the accuracy of an archive when the collection procedure is delegated to a context that does not match the archivist's perspective. Section III.3 emphasized that merely backing up data is insufficient for the preserved output to be considered an archive. Section III.4 suggested some methods to retain privacy on personal web archives collected and to ensure that if sensitive data is preserved, it is not trivial to reinterpret by parties with malicious intent. Section III.4.1 explored further as to why secure approaches are not accessible to casual users and thus a balance should be made to obtain a level of security to which the user feels is necessary to protect the archived data. Section III.4.2 went to the other extreme in analyzing why server-based solutions are not appropriate for personal web archives beyond the context of privacy and security. Lastly, Section III.4.3 discussed why archive integrity is important for preservation, especially for the realm of personal web archives.

CHAPTER IV

NEW TOOLS FOR PERSONAL WEB ARCHIVING

As described in Chapter II, previous approaches to archiving personal content on social media sites are not optimal in terms of retaining the look and feel of the original content and producing an archive in a standard format, such as WARC. In this thesis, I use two new tools, WARCreate and ArchiveFacebook, to demonstrate my approach to personal web archiving. I developed WARCreate for this thesis to archive any viewable web page into the standard WARC format. I contributed to the development of ArchiveFacebook, which backs up a user's Facebook pages and retains the look and feel of Facebook. Later in this thesis, I will use a modified version of ArchiveFacebook to demonstrate that using the proposed specification allows a tool to adapt to changes in a social media site's hierarchy. In addition, in this chapter, I will describe a modified version of the XAMPP client-side server suite that allows users to view the WARCs created by WARCreate in a local instance of wayback. Here I hope to enumerate the advantages and shortcoming of the extensions used in this thesis and how the supplementary server suite can assist in them achieving their goals.

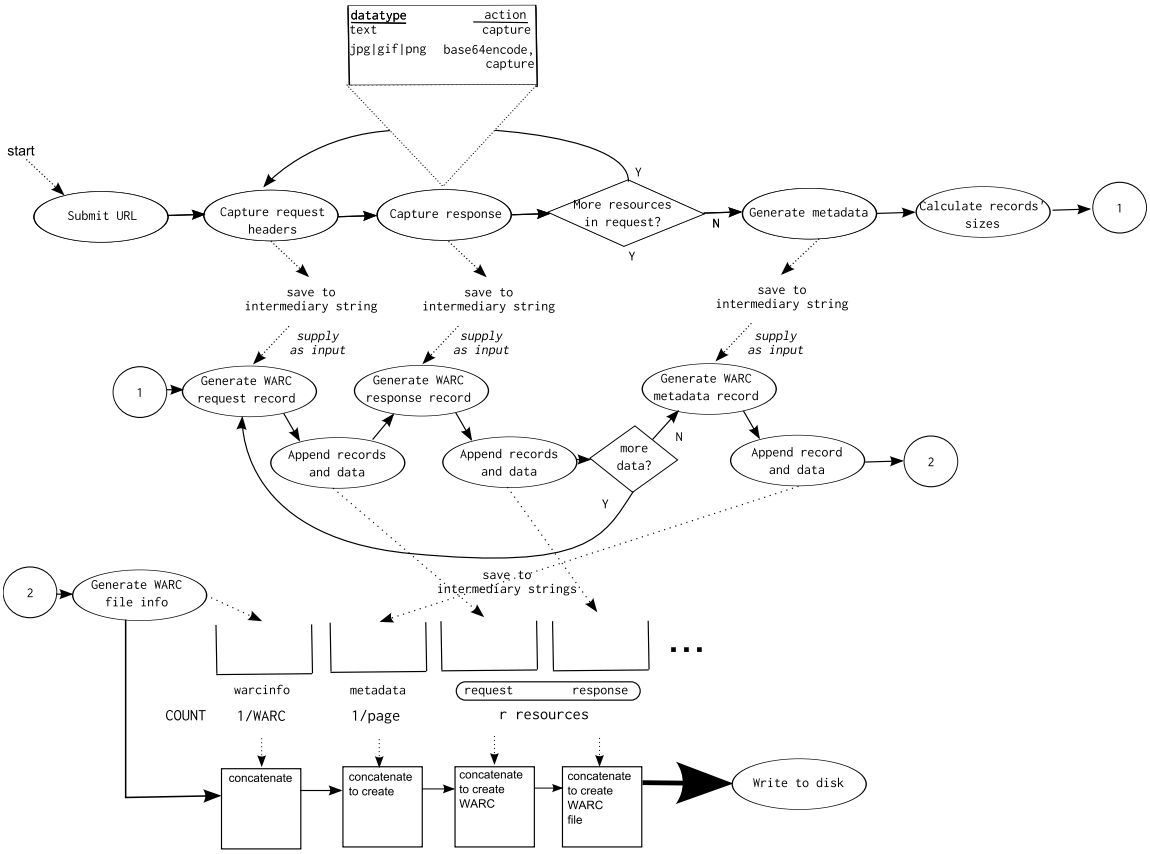


Fig. 11. WARCreate's operation relies on a sequence of intermediary storage because of the importance of content-length being explicitly defined for the WARC records and the payload. This sequence also takes into account the need to convert non-textual media to a form that can stored as text, namely, the media's base64 encoding.

IV.1 WARCREATE

WARCreate [31], a tool developed for this thesis, is an extension for the Google Chrome web browser that allows a user to generate a WARC file from the current webpage. To do this, the user clicks on the browser extension's icon in the address bar then presses the Generate WARC button. The browser extension gather the resources (including external scripts, CSS and images) and HTTP headers normally used by the web browser to generate a webpage and adds metadata (the *warcinfo* records) to generate a WARC file that conforms to the WARC standard's specification. It is this adherence to the specification that allows the WARC file to be read by Wayback.

The internal workings of WARCreate consist of a "collect", "concatenate", and "generate" series of operations (Figure 11) to produce a WARC file. When a page is

visited, even before the extension is instructed to preserve the content, the extension stores all HTTP headers that the browser sent and received into a collection of strings. If the extension is never given the command to generate a WARC from the page, this information is discarded. If the command to preserve is given, the extension collects all textual and binary content into strings as well. From the headers, metadata that is representative of the resources, as well as information about the archiving session being performed, is generated. The HTTP header content previously retained is attributed to each resource's representation as a string and concatenated along with the representation's respective metadata to produce a string representative of the preserved page. A record is then generated by WARCreate to provide metadata about the archiving session and is prepended to the archive to complete the generation of the WARC file. This file is then served to the user.

The model used by WARCreate can be applied to other tools. By leveraging the user's perspective of the web browser (Figure 12, marker 1), a user interacts with a tool (marker 2) that serves as a bridge for converting viewed content into an archived form. In the case of WARCreate, the process described in Figure 11 is executed (marker 3) and the extension outputs the file representing the archived content (marker 4) into a local repository. The concrete example of WARCreate outputting WARCs couples with the consumption of this archived format (marker 5) by a system created to read the format, a local instance of wayback separate from WARCreate. The user can then access this local instance (marker 6) and view the result of the processing instigated by the initial interaction (marker 2) of the tool (e.g., WARCreate) via the browser.

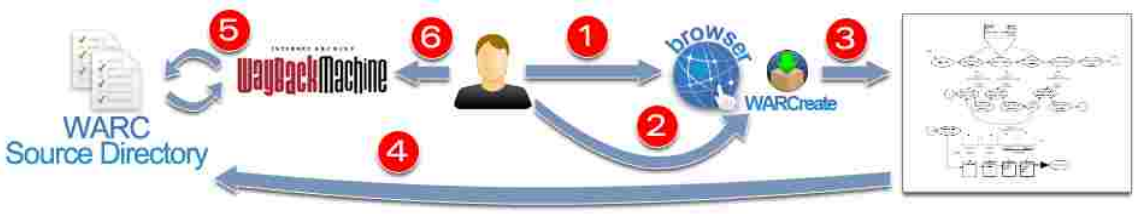


Fig. 12. A higher level view of an archival tool built upon the browser platform gives perspective on how all of the components of archive creation, consumption and replay can be experienced by the user. Displayed here is the process that WARCreate uses to produce a WARC file. The process is abstract enough for any browser-based tool to reuse by putting in-place its logic where WARCreate's logic currently resides (after marker 3).

The intention of the creation of the WARCreate Google Chrome extension was to sacrifice immediate viewability of an archive for the advantage of appealing to a standardized archiving format. WARC files are normally generated by the Heritrix web crawler, to be consumed by the Wayback Machine. With the assumption that a user would be able to leverage the open source nature of a tool (i.e., wayback) that was created to consume a robust and extensible format of digital archive (i.e., the WARC format), WARCreate generates this format of archive from any arbitrary web page and works toward bridging the gap that currently exists between institutional web archiving and personal web archiving.

WARCreate was developed for this thesis with one objective: to allow a user to save a webpage and all other required metadata (e.g., header information) to a WARC file. Its procedures, unlike Archive Facebook's, are generalized enough to be applicable to any webpage. WARCreate is not tied to any specific website's hierarchy, so it is immune to this sort of breaking. At the same time, WARCreate's website agnosticism prevents it from creating comprehensive WARCs cohesive of social media sites' content, like Archive Facebook.

To remedy this shortcoming, some form of sequential archiving (Section VI.2.1) as well as the ability to associate subsequent pages together to form a comprehensive (qua Archive Facebook) archive would a step forward for the tool. The implementation would need to stress the retention of the site agnosticism feature so as to not succumb to the breaking caveat of other tools. Once these shortcomings are addressed, having the tool conform to the specification will result in a tool that can

reliably create WARC files of all of a user's pages on a social media website.

IV.2 ARCHIVE FACEBOOK

Archive Facebook, originally developed by the Web Sciences and Digital Libraries (WS-DL) Research Group at Old Dominion University, operates by appealing directly to the breakup of content sections on the social media website Facebook.com. Though archivists have made previous strides in preserving websites like YouTube [11] and MySpace [18], a growing amount of personal (and what will be historically significant) information is locked behind the walled garden of Faceboook [51]. Through a system of scraping, the Firefox add-on is able to capture the content in the viewport of the user's browser to disk and resolve references to all downloaded resources. The final steps of the tool's operation link all of the sections together and provide a starting point for a user to replay the archive (see Figure 13) via an entry in the sidebar of the user's browser.

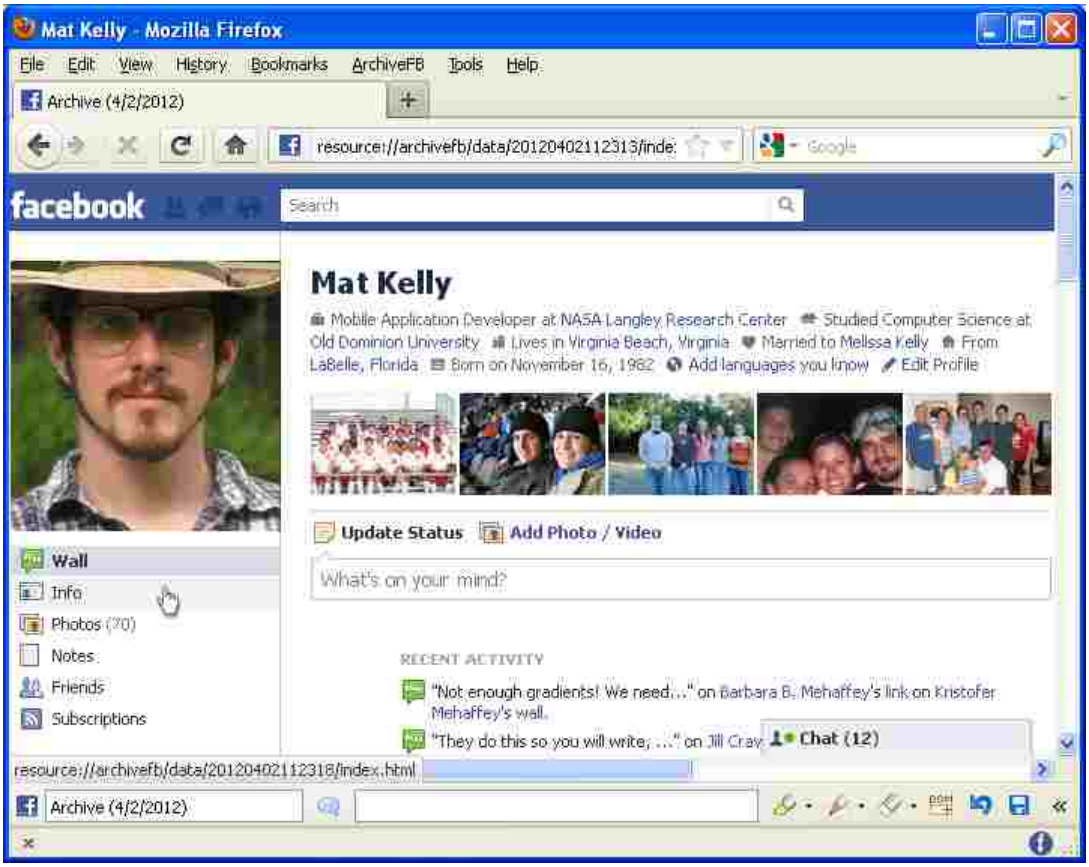


Fig. 13. Archive Facebook saves the resources it "archives" to the local file system, shown here as a navigable system of webpages linked with resource:// URIs. The add-on rewrites URIs that would normally point to the absolutely defined http://facebook.com resource and instead resolves them to local resources.

Though both WARCreate and Archive Facebook circumvent the issue of context and authentication, they do so in different ad hoc fashions. Archive Facebook does not have relevance to websites outside of Facebook, and WARCreate is merely a general purpose bridge of getting content from one form to another. Both of these tools only work in the browser for which they were respectively created. Through my work on both of these projects and the lack of extensibility that comes from these types of one-off tools, I am providing a standard for addressing the hierarchy and means for capturing content on social websites residing behind authentication, so that this content can be captured and treated in the same way and with similar tools as conventional web archives.

Archive Facebook reliably backs up (not archives, see Section III.3) the content of a user's Facebook profile. Because Archive Facebook is written to scrape the design

and hierarchy of Facebook at the time of its last update, its pattern-matching algorithm breaks upon Facebook's redesign or hierarchy change (resolved by conforming to the specification as documented in Section VI.2.1). A further shortcoming of the tool is that its output format is sub-optimal for archiving. Little information is retained in a usable format about the "archiving" operation that is performed when a user executes the tool's main process. Improvement of retaining this metadata would be a positive step toward the tool producing an archive.

The larger problem with Archive Facebook (aside from its inadaptability) is the output format. The product of Archive Facebook is a navigable directory of archived webpages. This is different than the product of a Facebook Data Dump (Section II.4.1) in that Archive Facebook preserves the look-and-feel of the original experience of the webpage and does not selectively exclude content based on the opinion of a third party. The latter is likely the case with the Facebook Data Dump because of content ownership, privacy and other concerns. Attempts in Section VI.2 improve on the tool's inadaptability issue, however, the output is still not in a format that portable, suitable for replay and integrates with other archiving technologies. Attaining these traits would require internal re-packaging or post-processing of the output. Doing this would make the tool more unwieldy for the casual user and would be akin to appending the entire functionality of WARCreate onto Archive Facebook, an endeavor that would require porting (from the Chrome API to the Mozilla API) as well the introduction of scope creep into the software's objective.

IV.3 RE-PACKAGED XAMPP

In Chapter II I emphasized that the goal of this thesis is to provide a way for personal web archiving tools to be adaptive. Rather than limiting the applicability of this thesis's work to tools that output to a subpar archiving format, I created WARCreate to bring personal web archiving one step closer to conventional web archiving by enabling users to preserve content into the WARC format. The WARC format is not meant to be consumed by the user but rather to be run through another medium for reinterpretation.

WARC files have little practical use if it is difficult for end-users to replay the archived content contained within. To validate that the content contained within the WARC files produced by WARCreate consists of all of the desired content, some wayback instance is needed, at least for visual validation of correctness. Because

WARCreate's objective is to make the WARC format more accessible for the end user in the context of web archiving, a preliminary implementation of an instance of wayback was also created so that the end product of WARCreate could be evaluated.

This implementation uses the software package XAMPP[1] (Figure 14) as a basis for providing the foundation for a local wayback install. XAMPP provides a portable implementation of its system, which allows the software package to be used without needing to be installed into the registry of the operating system. For this evaluation, XAMPP Portable was used on a Windows machine, but XAMPP is cross-platform so the choice of operating system is not a limitation. XAMPP Portable does not come stock with an Apache Tomcat[2] instance, which is required for the Java-based wayback package; however, the purpose of excluding the package by the developers was to make XAMPP Portable lightweight, and Tomcat 6 is easily installed through an Apache-supplied module for XAMPP.

[1] http://www.apachefriends.org/en/xampp.html
[2] http://tomcat.apache.org/

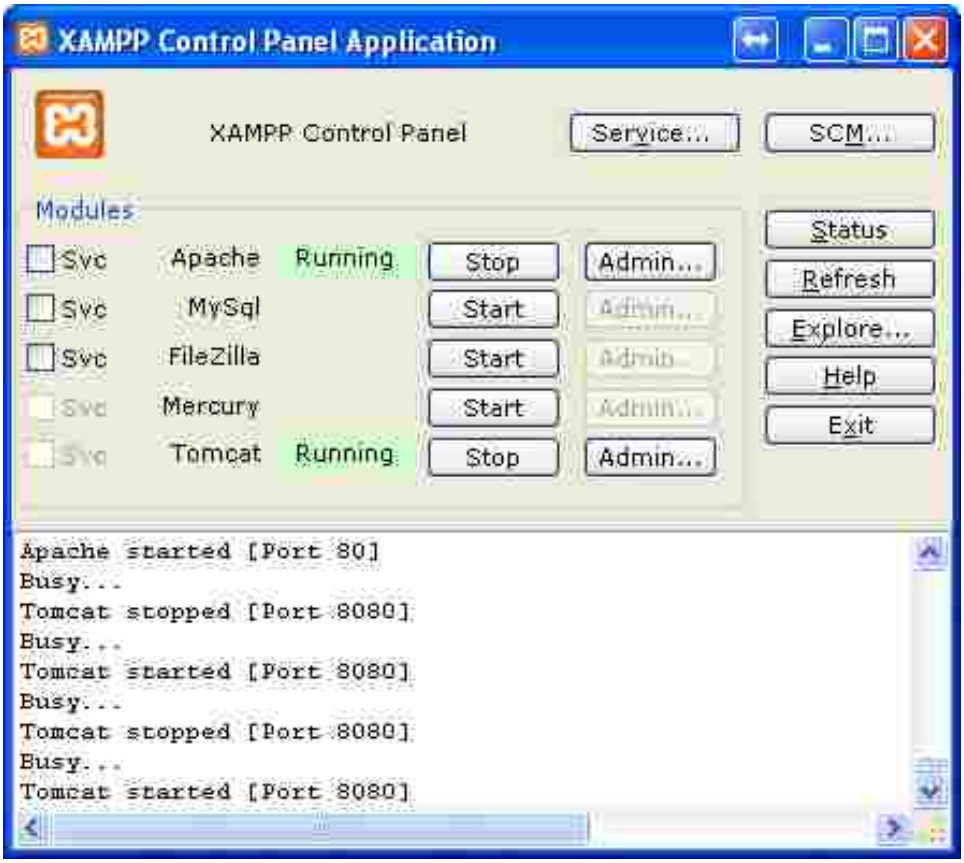


Fig. 14. The XAMPP package provides an easy-to-use interface to encourage the utilization of various packages created by the Apache Foundation. Shown here are the two services needed for the executing of the local wayback instance - Tomcat and Apache.

To encourage the use of WARCreate, I have repackaged XAMPP Portable to include the previously excluded Tomcat module. I also included wayback 1.6 in the package to be loaded when Tomcat starts. Further customization of wayback's configuration files has been specified to provide the source of WARC files to be within the XAMPP Portable folder, retaining the product's core package (XAMPP) encouragement of portability. This package is utilized by WARCreate by a user first navigating Google Chrome to a webpage they would like to be archived. The user then selects the button in the Chrome extension to begin the WARC creation process. The file is then downloaded to the user's system. Though the eventual workflow of the browser extension will make this process more seamless, the user then needs to move the downloaded WARC file to the location where wayback can find it for replay, which is specified in the modified wayback configuration and documented in

the repackaged XAMPP Portable.

A user is then able to navigate their web browser (any browser on the system, as the process is relieved of its coupling after the WARC creation process) to http://localhost:8080/wayback (or simply http://localhost:8080 in a future implementation of the repackaged product), search for the the archived URI in the text field and view the archived webpage just as they would when using the Wayback Machine at archive.org. It is not the objective of this thesis to merely demonstrate that content behind authentication can be archived. Regardless, validating that it *can* be archived using browser-based tools is important. These personal web archiving tools should result in a standard format (i.e., WARC) and put emphasis on retaining the user's original perspective (i.e., the web browser) when preserving content of a personal nature on the web.

IV.4 SUMMARY

Chapter IV describes two browser-based tools created to accomplish personal web archiving. WARCreate, a tool developed for this thesis, is described in Section IV.1 while a pre-existing personal web archiving tool, Archive Facebook, is explained in Section IV.2. Section IV.3 considered using a client-side server suite, XAMPP, to allow browser-based archiving tools to leverage server-side technologies.

CHAPTER V

CONSTRUCTING A GENERAL SPECIFICATION FOR SOCIAL MEDIA WEBSITES

In this chapter I will progressively build a specification to be used by archiving tools to become adaptive to the frequently changing design of their target websites. This will be done in a manner that justifies the end-result by alluding to the common hierarchical trait between various social media websites. Table I illustrates this commonality between Facebook, Google+, and Twitter. Here, I have abstracted seven of the most common social media site sections and shown how each of them maps to these popular sites.

TABLE I

SIMILAR ABSTRACTIONS OF RESOURCES EXIST ON NUMEROUS WEBSITES THOUGH EACH IS IMPLEMENTATION-SPECIFIC, WHICH CAN REQUIRE SUBCLASSING TO ACCURATELY DESCRIBE THE WEBSITE'S SECTION'S WORKINGS IN A CLASS-LIKE HIERARCHY. FACEBOOK'S "FRIENDS" MEDIA TYPE IS INHERENTLY BI-DIRECTIONAL, THAT IS, IF YOU HAVE A FRIEND, THAT FRIEND HAS YOU AS A FRIEND. IN GOOGLE+, RELATIONSHIPS CAN BE UNI-DIRECTIONAL. I CAN HAVE ALICE IN ONE OF MY CIRCLES BUT THAT DOES NOT NECESSARILY IMPLY THAT ALICE HAS ME IN ONE OF HERS.

Abstracted media type	Facebook	Google+	Twitter
personal stream	wall	posts	my tweets
global stream	news feed	streams	followees' tweets
multimedia - photos	photos	photos	*N/A*
multimedia - videos	videos	videos	*N/A*
photo collection	albums	*N/A*	*N/A*
posts	notes	*N/A*	*N/A*
friends	friends	circles	following

Creating a specification for addressing the problems to which tools like Archive Facebook and WARCreate eventually succumb is initially a design problem. Guidelines help to establish boundaries in situations where no collection policy exists [28]. Much like the WARC format, inherent extensibility should be a core part of guidelines that are setup to remedy these tools' problems. An allusion to Object Oriented Programming (OOP) is useful here, though the advantage of using it is not immediately apparent until a hierarchy is built and potential design problems are considered. Here I hope to design the hierarchy of a schema on which to base the specification in the spirit of a class model. Inheritance will be duly utilized, as the sections and operations pertinent to social media websites tend to have inherent extensibility much like a class structure where certain operations and traits are attributed to representative sub-classes.

Fig. 15. The hierarchy of the BBC website's sports section easily resembles the one described. The parent to the sports section would be represented as the NewsWebsite object with the sports section's siblings being "News", "Weather", etc. as representation by the navigation at the top of the page.

Websites are inherently hierarchical, allowing a user to navigate to more specific topics as the user traverses down the tree of navigation. An example of navigating a website would be to first access the site's homepage, then a section of the

site (say, "sports" on a newspaper's website) then a subsection like "top stories" then a specific article (Figure 15). A class-like structure to represent each state with explicit nomenclature would be *NewsWebsite*, *NewsWebsite_Section_Sports*, *NewsWebsite_Section_Sports_TopStories* and *NewsWebsite_Section_Sports_Articles*, respectively. This sort of lateral relationship of classes (in terms of the hierarchy), multiple inheritance (e.g., *NewsWebsite_Section_Sports_Articles* might get special properties from a class that defines "top stories" and one that defines articles in general), encapsulation (e.g., *NewsWebsite_Section_Sports_TopStories* is a container for articles that might not always belong to the container) and implementation of abstractions into concrete classes (e.g., because *NewsWebsite_Section* might be too generic, it might be considered abstract and require an implementation like *NewsWebsite_Section_Sports*) fits naturally to a website with conventional navigation. Many social media websites follow this conventional navigation strategy.

```
1 SocialMediaWebsite class
2 - homepage : str
3 - sections : section[]
```

Fig. 16. The class-like definition of a social media website is simplistic so as to be applicable to a wide range of sites. Specific traits that are only applicable to a specific website could be created by subclassing this definition.

The first rather trivial starting point in developing this hierarchy is to correlate a website with a root class as a basis. This bare definition (Figure 16) of where the website resides and the breakup of content is initially sufficient. It is important to remember that extensibility should be emphasized to mimic the relations in practice. We will develop a class hierarchy that is semantic and allows dynamically defined sections (henceforth, objects of type SocialMediaWebsiteSection) to be attributed to a class. These sections will provide their own implementation of a set of operations as is appropriate to the respective section, allowing the pertinence of an operation to be defined in the concrete object rather than a common generic "Section" class. A rationale for this can be easily considered with comparing the expected functionality allowed in a photo album section versus a user biography section.

Next, we will define the SocialMediaWebsiteSection (Figure 17), providing an

abstraction that would be common to a social media website section with a default implementation (e.g., setting the name of the section upon creation) with the intention that these attributes might be overridden in the classes that extend SocialMediaWebsiteSection.

```
SocialMediaWebsiteSection class
- name : str
- url : str
- [preprocessor : SocialMediaPreprocessor]
```

Fig. 17. The definition for a section of a social media website contains only fundamental attributes: the name of the section and the referencing URL. An optional "preprocessor" attribute allow for the application of a webpage preprocessing procedure onto both the classes that extend from SocialMediaWebsiteSection as well as those that utilize the class directly because of a lack of need for section-specific attributes and procedures.

```
SocialMediaPreprocessor class
- timeBetweenFirings : int
- maxFirings : int
- conditionBeforeSubsequentFirings : SocialMediaPreprocessorCondition

SocialMediaWebsiteCollection class
- name : str
- ordered : bool
- items : SocialMediaWebsiteSectionItem[]

SocialMediaPreprocessorCondition class
```

Fig. 18. A preprocessor allows a webpage to be programmatically manipulated prior to performing some operation, in this case, archiving. The SocialMediaPreprocessorCondition allows the preprocessor to require a condition prior to execution. The maxFirings and timeBetweenFirings attributes allow for repeatability of the preprocessor's page manipulation action.

Getters and setters of these attributes are assumed to exist though are not important to define, as we will use this design in a medium where definition of these functions here would be moot. The preprocessor attribute being defined here was learned through experience with websites moving toward the loading of content on an asynchronous basis, often triggered by user interaction. These sort of preprocessors (class shown in Figure 18) exist in no particular section type and are common in a wide array of sections. Even with its general pertinence, preprocessors are more appropriate to some sections than others, so its definition by classes that extend SocialMediaWebsiteSection is optional. To display the robustness and validate that the construction of this hierarchy resemble real sites, an implementation of the definition of the Facebook website (as of early 2012) is shown in Figure 19. The complete hierarchy sufficient to represent a wide range of social media websites can be seen in Figure 20. The only further peculiarity in this diagram is in the definition of the SocialMediaWebsiteMultimedia class. Polymorphism, an attribute of OOP, is exercised here in that a multimedia collection expects either a further collection or a concrete multimedia object (e.g., photo or video) as a child. Both representations of the further collection and concrete object extend a common parent class, which allows the recursive requirement of the the multimedia collection's "children" attribute to be adequately fulfilled by either subclass.

```
SocialMediaWebsite facebook = new SocialMediaWebsite(homepage =>
  "http://www.facebook.com")
facebook->decorate([
  new SocialMediaWebsiteSectionPersonalStream(
    name => "Wall",
    url => "http://www.facebook.com/profile.php?sk=wall",
    preprocessor => new SocialMediaScrollPrepreprocessor(
      timeBetweenFirings => 0,
      maxFirings = 0,
      conditionBeforeSubsequentFirings = null
    )
  ),
  new SocialMediaWebsiteSectionUserInfo(
    name => "Info",
    url => "http://www.facebook.com/profile.php?sk=info"
  ),
  new SocialMediaWebsiteSectionMultimediaCollection(
    name => "Photos",
    url => "http://www.facebook.com/profile.php?sk=photos",
    proprocessor => new SocialMediaScrollPreprocessor(
      timeBetweenFirings => 0,
      maxFirings => 0,
      conditionBeforeSubsequentFirings = null
    )
  ),
  new SocialMediaWebsite(
    name => "Notes",
    url => "http://www.facebook.com/profile.php?sk=notes",
    preprocessor => new SocialMediaScrollPreprocessor(
      timeBetweenFirings => 0,
      maxFirings => 0,
      conditionBeforeSubsequentFirings = null
  )
]);
```

Fig. 19. A SocialMediaWebsite object can be decorated (in the spirit of Design Patterns [24]) to only contain the child objects that are pertinent to that website. Here, the sections of Facebook have been added as children to the parent SocialWebsiteWebsite object. Using this method allows for prototype-driven objectification of websites, aligning with Javascript's ability to extend objects in this way. Also interesting to note is the ability of section objects (here, the "Notes" section) to implement the general SocialMediaWebsiteSection object if they have no further functional require beyond what the class defines.

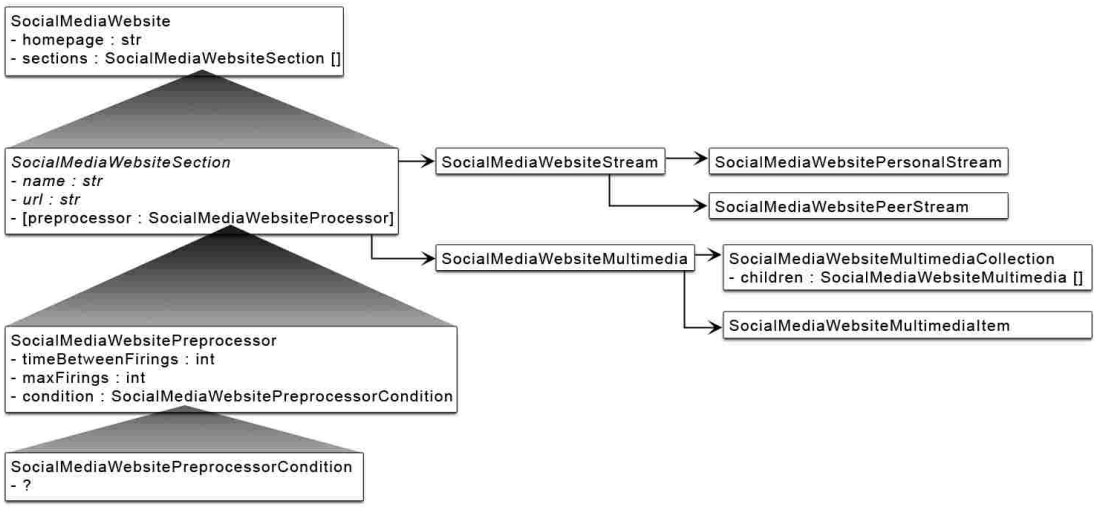


Fig. 20. While simple in definition, the inheritance chain of the defined classes that represent the different section types are sufficient for describing the hierarchy of many social media websites. Much of the power in this hierarchical chain comes from the common traits that many sections have and are defined in the abstract SocialMediaWebsiteSection class.

By defining this class hierarchy, we can convert our specification into an XSD specification. The specification can then be used as a basis for creating instantiations through definitions of social media websites that conform to the specification (see the correlative Facebook example in Appendix A). Externalizing the definition and site-specific specifications from tools allows tools to be adaptive as the websites change designs. The potential for breaking would be greatly reduced, as tools would be adaptive. Less time would then need to be spent on maintaining tools to correspond with target website design changes and instead focused on efforts of evolving the WARC format to appeal to facets of personal web archiving, especially for those websites whose content is not currently being preserved.

V.1 SUMMARY

Any means to access a webpage (e.g., browser, crawler, other scraping tool) should be able to parse out its contents or scrape out the data necessary from a webpage on a social media website's to determine the site's section breakup. Though the section

hierarchy of the target website may not be completely apparent from the navigational elements of a certain page, abstracting the breakup to an external document removes any complications that would arise in the site obfuscating the hierarchy. The web browser is particularly suitable for process this hierarchy because it, like other tools, contains the ability to fetch, cross-reference and manipulate data presented to it. Unlike crawlers and scraping tools, however, the web browser is the primary context in which the content is originally experienced. Its native support for XML and Javascript make it friendly to specifications of the type constructed in this chapter. As the specification is based on currently existing social media websites, its correctness can be tested, and this is done in Chapter VII.

CHAPTER VI

IMPLEMENTATION DETAILS

Two use cases previously mentioned will be discussed here to show that implementation of the specification is feasible if considerations of applying the abstraction are addressed. Each of Archive Facebook and WARCreate exhibit a lacking trait that causes the output to be less than ideal for what I hope to accomplish: to represent the content in a personal web archive accurately in the WARC format.

Archive Facebook suffers in that its output, while comprehensive, is in a format more similar to a conventional file structure and lacks much metadata. The former problem can be tackled by packaging resources in a way described by the WARC format while the former requires more data to be collected at the time of archiving.

WARCreate suffers from the opposite problem of Archive Facebook, so it works as a good use case in showing the robustness and wide applicability of the specification toward various software packages whose archiving procedure and output are sub-par. The primary downside of WARCreate is its limitation to a single web page and its inability to comprehensively create a WARC file that consists of a cohesive set of data representative of the pertinent content on the target website. This shortcoming will be addressed along with the concept of sequential archiving as is implemented in Archive Facebook. Unlike Archive Facebook's implementation, however, the implementation to be added to WARCreate will look to the guidance of the site-specific specification.

VI.1 USE CASE A: ADAPTING ARCHIVE FACEBOOK

The ideal archiving procedure to preserve social media websites is similar to that of Archive Facebook's in that it is comprehensive. A downside of Archive Facebook is that the preservation format is sub-optimal and an insufficient quantity of metadata is retained. Translating Archive Facebook to a better tool for archiving requires addressing the output issue as well as preserving metadata about the archive and converting the resources preserved into a more portable, self-contained format (i.e., the resource packaging problem). It is worthwhile to consider this tool further,

as it operates on a set list of URIs to poll and iteratively preserve (henceforth, "sequential archiving"). In this use case, I will discuss how this tool can be improved so its sequential archiving procedure can be leveraged and its shortcomings can be addressed and overcome. The improvements needed will be juxtaposed to the feature set initially programmed into WARCreate.

Archive Facebook does a good job of collecting all of the resources defined within the extension and writing them to the user's disk. What needs to be collected further to be able to create a WARC file from an archiving session initiated from Archive Facebook is to collect the HTTP headers for the requests of the resources, convert binary data collected to a portable form to be included in a WARC file, attribute the headers to the resources, generate information about the archiving session as WARC metadata and finally wrap all of the content together in a WARC file.

Converting the data to the appropriate form is only the first challenge in the task in allowing Archive Facebook originating archives to be able to take advantage of the specification. The URIs needed to successfully archive a user's profile, as defined in Archive Facebook, are hard-coded in the Firefox add-on, which is the reason for its breaking when Facebook changes its design or structure of resources.

Fig. 21. Archive Facebook allows users to specify which parts of their profile they would like archived. Each checkbox user interface element directly translates into a conditional clause in code containing the target UI representative of the section of the user's profile.

The order in which sections of Facebook are loaded and captured by the Firefox add-on is defined within the add-on. Conditionals are used (see Figure 21) for each section to determine if further processing is needed to capture all of the content,

and recursion is used where applicable. In the instance of a Facebook user's info, for example, two pages are collected: the user's wall and the "Info" section on the user's profile. The latter is merely a matter of loading the page and collecting all of the content, as before, except assuring that the headers are also captured. The former requires preprocessing before all of the content is displayed. The preprocessing needed, in this case, is to continue to scroll the page vertically until no further Ajax request is pending or until the threshold of unrolls, as defined by the user, has been met. This type of dynamic loading is common with social media websites that only load data when necessary to reduce wasted bandwidth and/or load times.

Other sections of Facebook require some form of recursion or consist of resources with more complicated data structures for which I have accounted in the class model. The Photos section consists of resources to be converted in its root as previews in addition to pages containing the full-size resource displayed and albums that can recursively be treated in the same way as the Photos section itself.

```
getCurrentSiteSpec : function(step,urlIn,hostIn){
 switch(step){
  case 0:
   var xhr = new XMLHttpRequest();
   var siteSpec = "", uriOut = "";
   $.ajax({
    url: urlIn,
    success: function(data){
     var host = "www.facebook.com"; //hostIn n/a here
     var parser = new DOMParser();
     var socialMediaWebsites = $(data.childNodes[0]).children();
     for(var i=0; i<socialMediaWebsites.length; i++){
      var smw = socialMediaWebsites[i];
      if($(smw).find("homepage").text().indexOf(host) != -1){
       siteSpec = $(smw).find("specification").text();
       getCurrentSiteSpec(1,siteSpec,host);
      } //fi
     } //rof
    }, error: function(){}
   }); //xaja
   break;
   case 1:
    $.ajax({
     url: urlIn,
     success: function(data){
      var ls = window.content.localStorage;
     ls.setItem("spec", (new XMLSerializer()).serializeToString(data)
        );
    archivefbBrowserOverlay.capture(ls.getItem("spec"));
   }, error : function(){}
   };
   break;
  }
}
var t = "http://www.facebook.com";
var protocolTrimmed = t.substr(t.indexOf("//")+2);
var host = protocolTrimmed.substr(0,protocolTrimmed.indexOf("/"));
getCurrentSiteSpec(0,"http://spec.socialstandard.org",host); //init
```

Fig. 22. The Ajax request for the specification file can neglect some of the edge case handling that would come about in needing to tailor the code to multiple browsers. Utilization of the jQuery library can be see in the general purpose $() selector function as well as in simplified iteration schemes.

Javascript normally has to be written to account for various browsers' quirks. Because the environment of execution will remain static in Mozilla (Archive Facebook is a Mozilla Firefox add-on), less overhead is needed in fetching the specification. Utilizing the jQuery[1] Javascript library, which would normally be a way to assure cross-browser compatibility, allows the adaptive code in Figure 22 to be simpler. Providing simpler examples of adapting tools will increase the appeal of conforming to the specification, as less overhead and a lower ad hoc learning curve will be needed.

VI.1.1 ADDRESSING THE RESOURCE PACKAGING PROBLEM

Acquiring representations of all of the resources used to generate a page is accomplished by querying the web page's document object model (DOM) once the page has been preprocessed to the degree specified, however this is problematic for resources that have already loaded. Often, because of security limitations, acquiring the encoded form of binary resources, like images, produces a security violation when the canvas-based approach show in Figure 23 is used.

```
1 function getBase64DataOf(img,imgType){
2   var canvas = document.createElement('canvas');
3   canvas.width = img.width;
4   canvas.height = img.height;
5   var context = canvas.getContext("2d");
6   context.drawImage(img,0,0);
7   return canvas.toDataURL("image/"+imgType);
8 }
```

Fig. 23. Binary data must be converted to an encoded form in order to store its contents inline with ASCII data. An HTML5 canvas-based approach works well for simple conversion but XSS concerns should be addressed is fetching and storing content across multiple domains.

Archiving tools that attempt to capture image content after the page has loaded will encounter a cross-site scripting (XSS) error when the the tool attempts to convert the image data from another domain to an encoded, binary form. The drawImage function at the end of the function in Figure 23 causes this violation. An alternative approach would be to capture the data prior to it being written to the DOM (which

[1]http://jquery.com/

would negate the requirement of converting to base64, as the data is already in that form) though the facilities to accomplish this are not necessarily accessible to a browser's extension API (e.g., Firefox's add-on API can do this but Chrome's extension API cannot).

VI.1.2 COLLECTING METADATA

While the Google Chrome API has the experimental webRequest[2], Mozilla's approach at retaining information beyond what is displayed and necessary for a WARC file (namely, the HTTP headers) is retained through the visitRequestHeaders() and visitResponseHeaders() of the nsIHttpChannel interface[3]. Further metadata can be generated using a templating system consisting of time of archive, derived content length of generated records and a UUID string to uniquely identify records and attribute records to one another.

A warcinfo record describes the file to be generated. An example template for a warcinfo record that complies with the WARC specification is shown in Figure 24. Further, a WARC metadata record, which describes a set of WARC records and not the WARC file itself, can use a template like in Figure 25. A 'metadata' record contains content created in order to further describe, explain, or accompany a harvested resource and will almost always refer to another record of another type, with that other record holding original harvested or transformed content [35]. Because of this, retaining perspective data in this record type would be the appropriate place to preserve it in a generated WARC file.

[2]http://code.google.com/chrome/extensions/webRequest.html
[3]https://developer.mozilla.org/en/XPCOM_Interface_Reference/nsIHttpChannel

```
1  WARC/1.0
2  WARC-Type: warcinfo
3  WARC-Date: {ISO8601 formatted time}
4  WARC-Filename: {generate warc filename}
5  WARC-Record-ID: <urn:uuid:{unique id}>
6  Content-Type: application/warc-fields
7  Content-Length: {length of following record}
8
9  software: {archiving tool information}
10 format: WARC File Format 1.0aco
11 conformsTo: http://bibnum.bnf.fr/WARC/
       WARC_ISO_28500_version1_latestdraft.pdf
12 http-header-user-agent: {user-agent information}
```

Fig. 24. An example warcinfo record describes the WARC file itself in contrast to all of the other records in a warcfile describe contents of the archive or metadata for other records.

```
1  WARC/1.0
2  WARC-Type: metadata
3  WARC-Target-URI: {target URI}
4  WARC-Date: {ISO8601 formatted time}
5  WARC-Concurrent-To: <urn:uuid:{collective identifier between all
       records representing this resource}>
6  WARC-Record-ID: <urn:uuid:{unique identifier attacked to this
       metadata record}>
7  Content-Type: application/warc-fields
8  Content-Length: {length of meta info below}
9
10 outlink: {reference URI for wayback to quick attribute the resources}
```

Fig. 25. Internally, WARCreate is template driven. WARC data that relies on the context of the target page is captured as appropriate. WARC data that is normally generated by the capture tool, (e.g., Heritrix) is fabricated by WARCreate. The crux of WARCreate lies in ensuring that all data in the records that consist of fabricated identifiers and experienced data are aggregated correctly to produce a valid WARC file.

VI.2 USE CASE B: ADAPTING WARCREATE

Unlike ArchiveFacebook, WARCreate was created to archive a single webpage and all of the resources to correctly re-display the webpage, including HTTP headers, into a single WARC file. The Chrome extension also provides a way to append new content onto an existing WARC file. For WARCreate to require all content in the generated or manipulated file to have the same domain origin would be against the nature of the WARC files, which frequently are very large in size in the use cases of Archive-It and the Internet Archive, and so likely contain archived pages from various domains. The extension does not provide the facilities of archiving all content associated with a user on a social media website without manually first loading each webpage. Still, the archive is only as comprehensive as Archive Facebook (which outputs the correct amount of content but in the wrong form) if the user visits every

page and performs the "Create WARC" procedure.

VI.2.1 SEQUENTIAL ARCHIVING

A sequential model similar to what Archive Facebook performs can be used as a basis for the procedure needed in applying the "archive whole website" or "cohesive website archiving" concept but abstracting the procedure to be applicable to all websites defined as instantiations of the specification. The initial approach is naïve about URL ordering [15] for simplicity, as a website-specific study would have to be done to prioritize target resources. Conventional methods of establishing priority [64] are likely not applicable due to the unique nature of pages in the Deep Web. Most web crawler strategies download the most important pages or retrieve the most frequently changed pages [64]. Facebook consists of a limited number of sections that are attributed to a particular user. Retaining this listing allows each to be accessed, pre-processed and captured iteratively until the list has been exhausted [15]. Though the sections could likely be executed concurrently with this process, it is in the nature of a website to block a large number of requests that occur at one time due to the potential of the requests being the start of a denial of service (DoS) attack. Opting for a sequential approach assures that the crawl is polite and avoids unduly high load on the target site's server [48]. Another concern is that, because of the controversial nature of data ownership [46,47], the website will block this sequence of page accesses if it happens from a specific source frequently. A website will use this method in an attempt to prevent this type of automated querying, which are likely violates the respective website's terms of service. As an example, Facebook's terms state, "You will not collect users' content or information, or otherwise access Facebook, using automated means (such as harvesting bots, robots, spiders, or scrapers) without our permission" [21]. Considering this further, a page access might have side effects on that page, causing another page's contents to be modified if, say, the page accessed second in a series has a, "pages recently visited" navigational item. These concerns are out of the scope of this research, as the primary objective is to capture this content. The latter problem does not compromise the archive integrity but shows that sequential archiving is problematic because of said side effects. Overcoming this problem would require techniques attributed to debugging self-modifying code.

VI.2.2 PRE-PROCESSING AND LIMITATIONS OF CONVENTIONAL

CRAWLING TECHNIQUES

A conventional approach, as is seen by Googlebot and Heritrix alike, is to gather all links on a webpage while potentially filtering for those within the same domain, set of pertinent sub-domains or some other cohesion metric [15]. A limitation to crawlers' approach is overcome in the pre-processing concept that is present in the specification defined in this thesis. Archive Facebook leverages this concept in its loop unrolling. The gist of this procedure is to perform an action on a page to cause all desired content to be displayed prior to initiating the archiving procedure. This pre-processing varies between websites, can be changed over time by the service and is not usually performed by web crawlers because of its ad hoc relevance to a particular webpage. WARCreate was originally designed simply to capture the content behind authentication into the WARC format and does not do any such unrolling like Archive Facebook. The specification addresses these sort of procedures that are relevant to any webpage that dynamically loads content though is particularly useful in social media webpages that benefit greatly from only loading content as-needed because of the volume of users served.

WARCreate currently does not pull an entire website into a WARC file but it could if it had a sequence of sections and operations to be performed on a target website, which is what the spec would provide. The Chrome Extension Tabs[4] module provides the facilities to load new URIs. In conjunction with the webRequest module, new pages that are loaded can have their headers retained and attributed to the resulting content once the load operation has finished. A restriction on synchronicity exists in attributing the headers retrieved with the resulting content. The simplest approach is to implement sequential archiving (Figure 26): loading a page, capturing the headers, generating the metadata then combining subsequent repetitions of this process while adjusting the warcinfo metadata appropriately if to be combined into a single file.

[4]http://code.google.com/chrome/extensions/tabs.html

```
1  var requestHeaders = [], responseHeaders = [];
2
3  function sequentialArchive(urisFromSpec}{
4    var dataStr = "";
5    foreach(uri in urisFromSpec){
6      chrome.tabs.create({url:uri});
7      dataStr += createWARCStringFrom(dom.content,requestHeaders,
           responseHeaders);
8      // ^ pseudocode to abbreviate actual WARCreate implementation,
           which is lengthy and complex
9      requestHeaders = []; responseHeaders = [];
10   }
11   dataStr = prependWARCInfoRecord(dataStr) + dataStr;
12   return dataStr;
13 }
14
15 chrome.webRequest.onSendHeaders.addListener(function(){
16   //logic to capture request headers
17   //populate requestHeaders array
18 }{ urls: ['http://*/*'] }, ['requestHeaders','blocking']);
19
20 chrome.webRequest.onResponseStarted.addListener(function(){
21   //logic to capture response headers
22   //populate responseHeaders array
23 }{ urls: ['http://*/*'] }, ['responseHeaders']);
```

Fig. 26. A single iterative loop utilizing the Chrome Extension API is sufficient for implementing sequential archiving into the tool. A more ideal approach would be to nest a second level of indirection into the associative arrays representing the headers. The first level's key would be the URI and the value another associative array with each key being the header name. This would allow a more concurrent approach at archiving to be used but for the sake of simplicity, a more rudimentary set-then-clear sequence was used to demonstrate the implementation.

VI.3 IMPLEMENTATION-SPECIFIC CAVEATS

The browser extension/add-on architecture of the Google Chrome and Firefox have built-in addressing of security concerns that arose as the browsers evolved.

Each browser extension is primarily written in Javascript and because of the limitations of the allowed scope of the language (in regard to interacting with the user's file system), the implementation and further maintenance of the extensions is becoming more browser-specific with time. APIs common to both the Chrome extension and Firefox add-on APIs (e.g., NPAPI[5]) are helping standardize the methods needed to implement a common feature-set across browsers. However, the independent development of each browser and the competitive need of each browser to offer features not present in others is causing the intersection of a common API to shrink as each browser further matures.

VI.3.1 INTERACTING WITH THE FILE SYSTEM

The method that Archive Facebook uses does not allow files to be written to a chosen location on disk but rather must reside in a specific "sandboxed" location, as enforced by the browser[6]. While developing WARCreate in support of this research, an alternative method of allowing a user to interact with the local file system, particularly to save WARC files to disk, needed to be explored. HTML5's File API [59] is implemented almost in full in each of Firefox and Chrome at the time of this writing, though it is insufficient for the degree of file system interaction required. The file system exposed to the user in the HTML5 file system API is sandboxed away from the conventional file system, preventing the user from storing generated WARCs in an arbitrary folder on disk that is specific for WARC processing by a local wayback instance.

WARCreate interfaces with the currently displayed webpage by capturing the HTML, HTTP headers and images as Javascript strings, with the latter being first converted to base64 binary data for portability. Javascript is unable to write strings to an un-sandboxed part of the user's disk and (as of this writing) cannot provide a "Save as" dialog to the user to save the string as a file to disk. Ways to overcome this limitation in the implementation of the language in Google Chrome proved fruitless, so a server-based approach was used instead with the hope that the HTML5 API provides a means in the future. The server-side approach consisted of sending the Javascript string data to a script on a server that then processes the data and serves it back as a file with the correct content types specified in the HTTP headers.

[5]https://wiki.mozilla.org/NPAPI

[6]https://developer.mozilla.org/en/Code_snippets:File_I/O

The server-side approach comes with advantages and disadvantages. Firstly, because the browser extension is reliant on a server-side script, a single point of failure is introduced. Further, the requirement to create WARCs from any arbitrarily webpage is no longer met, as machines that are not connected to the Internet (e.g., a user trying to create a WARC from an intranet page while not having Internet access) are no longer capable of accomplishing the sole purpose of the extension.

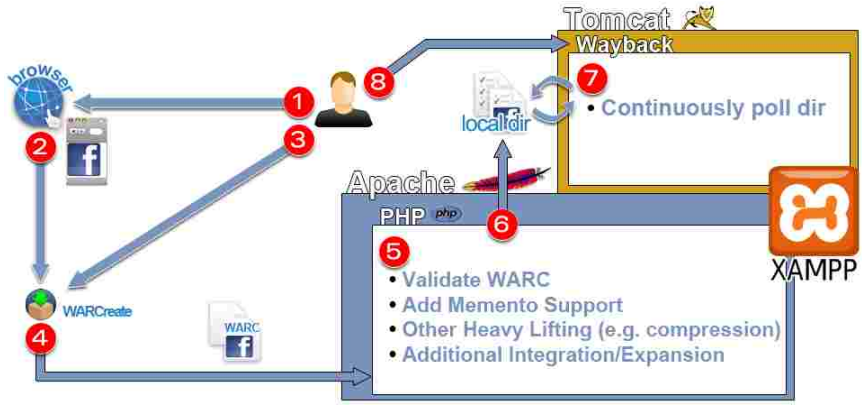


Fig. 27. Utilizing technologies that are more fit for a server than a user's machine does not necessarily imply that a remote machine must be used. Some of the difficulties of interacting with the file system are overcome by providing server-like functionality onto a user's machine. XAMPP, a package suitable to accomplish this, allows just this and is discussed more in Section IV.3. By utilizing their web browser (marker 1), a user allows WARCreate to capture the HTTP headers of a browsed webpage (marker 2) and optionally tell the Chrome extension to generate a WARC from this page (marker 3). The extension sends this WARC to a localized server (marker 4) to be validated, integrated with other technologies and optimized (marker 5) and saves it to a local directory (marker 6). This directory is accessible to the user's local Wayback instance to have its contents indexed (marker 7) and served through replay to the user (marker 8).

Going the server-side route (Figure 27) provides some advantages that far outweighed the above limitations. As discussed in Sections III.4.1 and III.4.2, a decentralized approach is ideal but requires a compromise in the security of the encoded data, as the performance hit is moved to the client. By accepting that a centralized approach is required due to the limitations in Javascript, we might also embrace the

centralized approach and integrate the encryption of the data with methods more conventional to enterprise web services. In the hope that the Chrome extension API and/or Javascript will eventually allow direct file system interaction or another secure means can be used, this higher degree of security has not been adopted simply to prevent too tight of a coupling with the server implementation.

Another opportunity that makes the server-side approach much more advantageous for the initial WARCreate implementation is WARC validation. While wayback provides some leeway in deviation from the WARC standard (it tries to recover if the files are not 100 percent compliant), there is no guarantee that other tools that implement the WARC standard will be as forgiving. WARCreate currently does not provide any way to ensure that its output complies with the WARC standard. A means to verify completeness and correctness is needed [67]. A way that an end-user of the open source wayback package could normally accomplish this is to run a program that is packaged with wayback, cdx-indexer, with the WARC file as input. Any errors in the file could be noted and the file manually repaired. This process is tedious and error-prone. My hope is to integrate cdx-indexer into WARCreate in the future but until then, validation of compliance of the generated output with the WARC standard is critical. By using the server-side approach, the data passed to the server can be validated and automatically repaired prior to being returned to the user to be saved to his file system. Further processing and analysis (e.g., WARC cohesion visualizations [66]) can be performed if the server-side approach is embraced.

Archive Facebook was originally developed based on the Mozilla Firefox extension Scrapbook [57]. Scrapbook [25] uses Javascript code modules to allow direct interaction with a user's file system thus allowing the downloaded resources to be directly written to the file system outside of the scope of the the HTML File API sandbox. Originally writing WARCreate for the Mozilla add-on API would have allowed a completed decentralized approach to be used but doing so might not have exposed some of the other shortcomings documented that are only pertinent to the Chrome extension API[7].

VI.3.2 LIMITATIONS OF THE EXTENSION API

As of this thesis' writing, the webRequest module of the Google Chrome Extension API (first mentioned in Section VI.1.2) is still fairly new and slightly problematic

[7]http://code.google.com/chrome/extensions/api_index.html

in retaining HTTP headers to record in WARC request and response records. For example, the documentation for webRequest[8] states "The following headers are currently not provided to the onBeforeSendHeaders event. This list is not guaranteed to be complete nor stable." The document then proceeds with a bulleted list of HTTP headers. Attempts at capturing the response headers received results in a similar subset of headers from webRequest that actually resided in the response and are subsequently handled by Chrome. Because that which can be captured is not representative nor comprehensive of what the user experienced, the WARC files generated from WARCreate suffer from the limitations of the API. With the rapid evolution of the Chrome extension API, however, this issue is likely to be resolved as the once experimental webRequest module matures.

The Chrome extension's offerings also suffer in that the raw data cannot be retained prior to being forwarded to the browser. Subsequent to loading, the DOM is captured by WARCreate. Though this allows user interactivity prior to capture, having the ability to capture data prior to being interpreted by the browser would simplify the extension's implementation and would allow for the capture of everything that is intended to be interpreted by the browser. Mozilla Firefox does not suffer from these issues with its more mature nsIHttpChannel[9] interface.

VI.4 SUMMARY

Chapter VI applied what had been built in previous chapters to show that the specification in this thesis is not only easy to implement but that current tools can be easily adapted to conform to the specification and receive the benefits it provides. Archive Facebook was adapted in Section VI.1 to utilize the specification with special concern toward some of the tool's shortcomings, namely its output format (Section VI.1.1) and its potential (yet non-utilized) opportunity to collect metadata (Section VI.1.2).

In Section VI.2, WARCreate was shown to be a better archiving tool when utilizing the specification and its primary shortcoming of lacking the ability to sequentially archive websites (Section VI.2.1) was highlighted but shown to easily be overcome. An advantage of WARCreate over Archive Facebook and the entirety of the class of crawlers was described in Section VI.2.2 where, by WARCreate being absolved of the

[8] http://code.google.com/chrome/extensions/webRequest.html
[9] https://developer.mozilla.org/en/NsIHttpChannel

sequential archiving functionality, it allows a user to manipulate a webpage's content prior to archiving. Because this is especially important in websites whose content may not be shown until a user interacts with it, this was a large contribution of the software package in capturing content as prescribed by the specification.

Section VI.3 considered personal web archiving from the browser as a whole and some additional problems it faces but will be overcome as technology evolves. The limitations of Javascript in interacting with the file system were described in Section VI.3.1. The current state of the Google Chrome extensions API was described in Section VI.3.2 by highlighting that, though the webRequest module of the API is not yet mature, the facilities to accomplish the task to which I originally employed the module already exists in other browsers.

CHAPTER VII

EVALUATION

In this chapter, I will evaluate the effectiveness of tools that use the proposed specification to preserve content behind authentication. To demonstrate the increased robustness that implementing the conformity to the specification into web archiving tools provides, a certain amount of naïveté on the part of the tools should be put in place so as to assure that that which should be evaluated is being evaluated. Aside from not being website agnostic and the output being more akin to a backup than an archiving procedure, Archive Facebook is the most suitable candidate of the software packages to be adapted to evaluate the result of conformity to the specifications. Per the table in Appendix B, WARCreate and "Save Webpage As" also appear to retain the traits that should be exhibited by personal web archiving tools, yet neither tools' encouragement of cohesion (established through sequential archiving as conveyed in Section VI.2.1) make them both sub-standard for testing the adaptability of a tool when the target site changes and the subsequently, the specification is adjusted.

Archive Facebook relies on a series of hard-coded Facebook-specific URIs to define the content that is to be preserved. In Section VI.1 I adapted the tool to make these URIs dynamic. Changing the source implementation of the specification and removing some Facebook-specific functionality from the tool will allow the save-to-disk capability of the tool to be utilized without deviating completely from the program's procedural flow.

This evaluation qualifies the success of the primary objective of instilling adaptability has been achieved through conforming to the specification.

VII.1 EXPERIMENTAL SETUP

I first setup a generic social media website at http://test.socialstandard.org for use in this phase of the evaluation. I will be investigating the degree of robustness of archiving tools when the hierarchy of the target website changes. This synthetic website consists of three sections, with one of the sections having two levels of additional depth in the hierarchy:

- **Peer Stream** - an amalgam of information and posts created by a user's peer on a social media website. It frequently resides at the target website's homepage (e.g., Facebook's "News Feed" at http://www.facebook.com). See Figure 28.
- **Personal Stream** - those posts and submissions created by the user and displayed on a single page. An example can be seen with Facebook's "Wall", residing at http://www.facebook.com/profile.php.
- **Photo Albums** - a page consisting of a means to reference other encapsulated resources exemplified by multimedia collections. For example, a user's photo albums can be accessed on Facebook at http://www.facebook.com/profile.php?sk=photos.

Fig. 28. The synthetic social media website setup for experimentation is database driven and consists of a hierarchy similar to conventional social media websites per Table I. Shown here is the aggregate feed of a user named Lorem Ipsum's "friends"' information temporally intertwined with his own posts.

To demonstrate the hierarchical section breakup that is common with social media websites, a **Photo Album** sub-section as well as a **Photo** sub-section will also be used. The general "photo albums" section on a social media website, in this case, consists of links to individual photo albums. Each Photo Album sub-section consists of links to photos.

A tool setup to archive the social media website at test.socialstandard.org would first reference spec.socialstandard.org, which contains an XML listing of all social media websites for which a specification exists. This definition would look similar to that shown in Figure 29.

```
<socialMediaWebsites>
  <socialMediaWebsite>
    <homepage>http://www.facebook.com</homepage>
    <specification>http://spec.socialstandard.org/facebook.xml</
       specification>
    <version>1.0</version>
  </socialMediaWebsite>
  <socialMediaWebsite>
    <homepage>http://test.socialstandard.com</homepage>
    <specification>http://spec.socialstandard.org/test.xml</
       specification>
    <version>1.0</version>
  </socialMediaWebsite>
...
</socialMediaWebsites>
```

Fig. 29. The root of the specification website contains an XML document that provides references to all of the site-specific specifications. Determining the applicable specification is as simple as first querying this document, matching up the target site to the "homepage" field and then acquiring the correct specification by fetching the subsequent XML document in the "specification" field.

The <homepage> tag serves as a reference for tools to use as a filter in acquiring the relevant specification definition. In the case of the previously defined test social media website, the definition of the section breakup resides at http://spec.socialstandard.org/test.xml.

Tools then access the location indicated in the <specification> tag to obtain an XML page with a definition describing the respective website's content hierarchy, as shown in Figure 30.

```
<socialMediaWebsite>
  <homepage>http://test.socialstandard.com</homepage>
  <sections>
    <socialMediaWebsiteSection
      type="SocialMediaWebsiteSectionPersonalStream">
      <name>Personal Stream</name>
      <url>http://test.socialstandard.org/personal</url>
      <preprocessor type="SocialMediaScrollPreprocessor">
        <timeBetweenFirings>0</timeBetweenFirings>
        <maxFirings>0</maxFirings>
        <conditionBeforeSubsequentFiring></
            conditionBeforeSubsequentFiring>
      </preprocessor>
    </socialMediaWebsiteSection>
    <socialMediaWebsiteSection
      type="SocialMediaWebsiteSectionMultimediaCollection">
      <name>Photo Albums</name>
      <url>http://test.socialstandard.org/albums</url>
      <preprocessor type="SocialMediaScrollPreprocessor">
        <timeBetweenFirings>0</timeBetweenFirings>
        <maxFirings>0</maxFirings>
        <conditionBeforeSubsequentFiring></
            conditionBeforeSubsequentFiring>
      </preprocessor>
    </socialMediaWebsiteSection>
    <socialMediaWebsiteSection
      type="SocialMediaWebsiteSectionPeerStream">
      <name>Peer Stream</name>
      <url>http://test.socialstandard.org/</url>
      <preprocessor type="SocialMediaScrollPreprocessor">
        <timeBetweenFirings>0</timeBetweenFirings>
        <maxFirings>0</maxFirings>
        <conditionBeforeSubsequentFiring></
            conditionBeforeSubsequentFiring>
      </preprocessor>
    </socialMediaWebsiteSection>
  </sections>
</socialMediaWebsite>
```

Fig. 30. The document at spec.socialstandard.org/test.xml contains the specification for the synthetic social media website created for this thesis.

VII.2 EXPERIMENTAL HYPOTHESIS

A tool configured to use a specification like the one in Figure 30 for archiving the contents of the target website (i.e., test.socialstandard.org) may or may not function when the target website's hierarchy changes. The tool's functionality will be restored to at least the same degree (potentially becoming aware of previously nonexistent sections) when the specification is updated to reflect the target website's new hierarchy.

VII.3 TOOL SELECTION TO VALIDATE POTENTIAL ADAPTABILITY

Integrating the specification with crawler-like tools may be inappropriate because pages that should not be included will be crawled. An example of this is exhibited in the hypothetical use of a crawler on Facebook, naively assuming that issues of perspective, authentication, user-interactivity, etc. do not inhibit its functionality. A crawler with a domain restriction would follow links unless otherwise directed not to do so (via the "nofollow" attribute). Without the domain restriction, the cohesion (Section VI.2.1) relative to the target website would be quickly broken. Without the intervention of the target website in directing the crawler through supplying nofollow to content that should not be archived, the crawler will add the discovered URIs to its list of URIs to crawl. Because crawlers would either break cohesion or archive beyond necessary scope (introducing excessive noise in the result), they are unsuitable for specification-driven archiving. Another type of tool should be chosen without these limitations, namely one that possesses a sequential archiving procedure.

To test whether a tool has become more adaptable to a website's hierarchy change after integration with the specification would be as simple as inducing a change in the Facebook website after having adapted Archive Facebook to use the specification (Section VI.1), a sort of Focused Crawling [14]. This is not possible from an end-user perspective and attempts to simulate this by URI rewriting, system-level hosts file[1] manipulation, or another means would leave open the question of whether the same functionality would work without this needed implementation prefacing.

I again modified Archive Facebook based on the version created in Section VI.1 to no longer reference the Facebook remote specification but instead to reference

[1]This file allows a system to artificially map any hostname, valid or otherwise, to a chosen IP address

the specification that corresponds to test.socialstandard.org. I also removed the ad hoc nature of Archive Facebook to allow it to be applicable to websites beyond its original intention. Other archiving tools (e.g., Heritrix, WARCreate) already meet this requirement of website agnosticism but do not execute using a sequential archiving (Section VI.2.1) scheme. They instead rely on a recursive crawling scheme (i.e., like Heritrix) or do not perform a series of iterative (e.g., processing a list of URIs) processing that would maximize the cohesion of the resulting URIs. Frequently, a list of URIs is provided and crawled with no guarantee that these URIs relate or are representative of a single website. Archive Facebook's ad hoc (in respect to the target website as explained in Section IV) iterative procedure is not coupled to Facebook URIs but instead to any arbitrary collection of URIs, be it defined within the add-on itself or remotely. Ironically enough[2], this makes Archive Facebook the most suitable tool in determining the success of applying the specification. The form of the preserved content can be neglected here as only the adaptability of the tool is being determined.

VII.4 PROCEDURE TO EVALUATE THE EFFECT OF A SOCIAL MEDIA WEBSITE'S CHANGE IN HIERARCHY

The simple social media website hierarchy described in Section VII.1 allows the base case of hierarchy change to be easily observable. I initially validated the functionality of the tools to ensure their capacity to create an archive with the changes needed to modify the tools to reference the specification.

The following are the steps required in performing the evaluation:

1. For a chosen archiving tool, run the primary archiving procedure with the target being a website where the site's hierarchy can be modified. This should result in an archive/backup of the site's contents, *BaseArchive*.

2. Modify the source of the tool to pull the URIs, which are hard-coded into the tool, instead from the specification with correct correlative substitution to the target website.

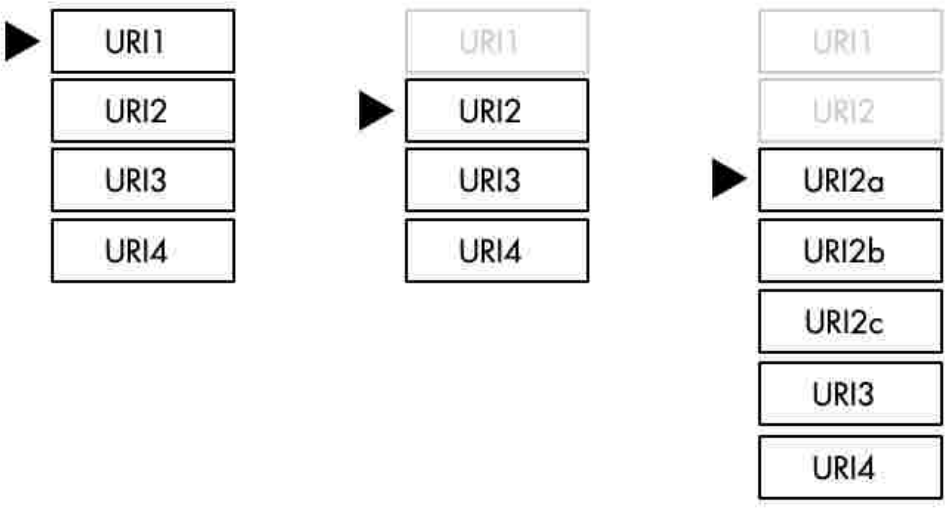
3. Re-run the procedure as in Step 1 to cause the primary archiving functionality to be performed, generating the output *ArchiveFromAdapted*.

[2]Ironic because Archive Facebook produces backups and not archives as in Section III.3.

4. Verify that *ArchiveFromAdapted* matches *BaseArchive*.
    - If results do not match, experimental setup was not performed correctly. This step serves as validation.
5. Modify the target website's hierarchy by performing a URI replacement change e.g., http://test.socialstandard.org/personal to http://test.socialstandard.org/myfeed
6. Perform Step 1 again, generating the output *IncompleteArchive*.
7. Compare *IncompleteArchive* to *ArchiveFromAdapted* and *BaseArchive*, noting the incomplete contents of the result.
8. Modify the specification of the hierarchy to represent the new structure of the target website by changing http://test.socialstandard.org/personal to http://test.socialstandard.org/myfeed
9. Again perform Step 1 to generate the output *ArchivedFromSpec*.
10. Compare the result of *ArchivedFromSpec* to *ArchiveFromAdapted* and *BaseArchive*. If the newly generated result matches those previously produced and declared correct, consider the tool adaptive.

VII.5 FROM ARCHIVE FACEBOOK TO COHESIVE SOCIAL MEDIA SITE BACKUP

Archive Facebook's primary interactivity code resides in the file overlay.js. When a user initiates the command for the add-on to start the archiving process, the user is presented with a DOM-based user interface (Figure 21 in Chapter V) generated by the add-on that allows the user to select which section of the user's profile the user wants archived. An anonymous Javascript function is tied to the "Begin Archiving" button that is selected after the user is satisfied with the options chosen in the generated UI. The function analyzes the selected options and iterates through a series of mapping a section (represented as a form selection in the generated UI) to a URI or regular expression based URI scheme (to account for dynamic data residing in a URI) that is defined within the add-on. Sections that have subsections are extracted through similar regular expression based scraping schemes and added to the front of the queue (Figure 31) of URIs to be processed.



(a) (b) (c)

Fig. 31. URIs are iteratively processed in a mutable queue (31a to 31b). When a URI is encountered that represents a section that contains subsections (e.g., "albums" section contains multiple "album" subsections abstractly shown as URI2 in 31b), the discovered URIs are placed at the front of the queue (31c) to be processed before URIs that were siblings to URI2. This process can be recursively repeated, essentially representing depth-first processing.

```
var capturedURIContentPairs = [];
function execCapture(uri){
  loadURI(uri);
  content = saveContentOnPage(uri);
  capturedURIContentPairs[uri] = content;
  subsectionsFound = findSubsectionsIn(content);
  foreach(subsection in subsectionsFound){
    execCapture(uri);
  }
}

foreach(sectionURI in websiteSections){
  execCapture(sectionURI);
}
convertAbsoluteToRelativeLinks(capturedURIContentPairs);
writeToSandboxedDiskspace(capturedURIContentPairs);
```

Fig. 32. Abstracting the Javascript code of the original Archive Facebook's into more generic pseudocode shows that its logic is generally applicable, even with hard-coded URIs. Note that Javascript's allowance of scope violation is exploited to retain a reference to all of the archived content and URI identifiers so that a cross-referencing URI-replacement scheme can be used to rewrite URIs that were absolute on the target pages to URIs that are local to the archive.

Figure 32 shows Javascript-like pseudocode of Archive Facebook's sequential archiving capture routine, which is representative of the graphical procedure of recursive queueing of Figure 31. Changing the mappings of the hard-coded URIs that would populate "websiteSections" set of URI strings in the code is a simple fetch, extract, and replace procedure. The crux of abstracting Facebook's section types into a descriptor that will be applicable to other sites, namely the synthetic social media website, is ensuring that the type of media to be archived and the respective hierarchical schemes are accurately represented. For the sake of simplifying the implementation, the step present in Archive Facebook wherein a user is given the option to exclude certain items from the archiving process was excluded from the adaptation of Archive Facebook to test.socialstandard.org. Table I from Chapter V can be modified into Table II to show where the two websites align in hierarchy.

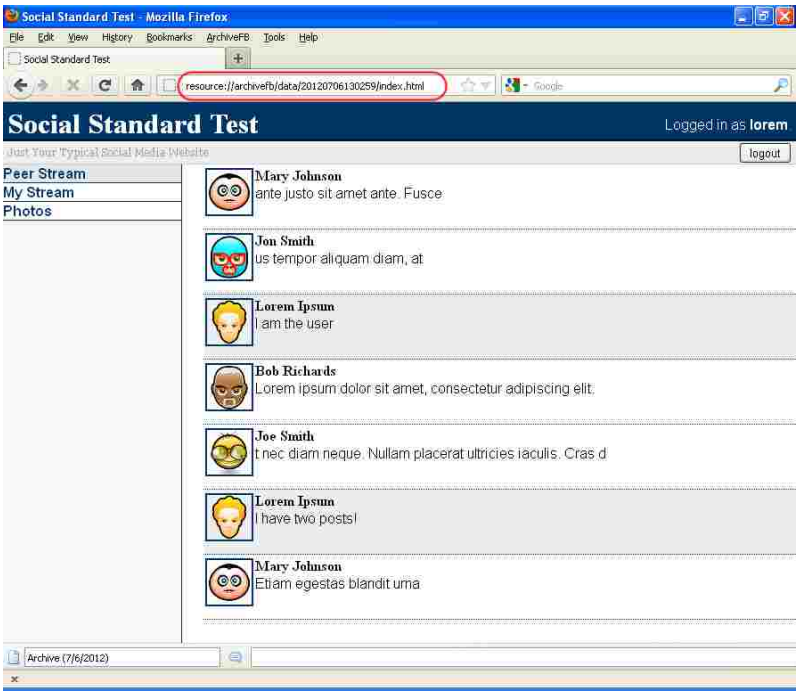
TABLE II

MUCH HAS BEEN STRIPPED AWAY TO REDUCE REDUNDANCY OF MEDIA TYPES THAT ARE SIMILAR.

Abstracted media type	Facebook	Test.SocialStandard
personal stream	wall	My Stream
global stream	news feed	Peer Stream
multimedia - photos	photos	Photos
multimedia - videos	videos	*N/A*
photo collection	albums	Albums
posts	notes	*N/A*
friends	friends	*N/A*

VII.6 RUNNING THE EXPERIMENT

From Step 1 in Section VII.4, the tools base implementation verified that with the current state of the target website, the tool is capable of archiving comprehensively. A page (part of *BaseArchive*) archived with the base implementation of Archive Facebook conforming to the test.socialstandard.org spec is shown in Figure 33a.



(a)

(b)

Fig. 33. A test-run of the tool to be used to show the instilled adaptability has resulted in this local copy of the test.socialstandard.org website (33a). This page is part of *BaseArchive*. The detail of the URI in 33b shows that this resource is locally stored as well as the timestamp representing the date of execution.

Step 2 requires the internal code of the tool to be modified to pull the target URIs from the specification instead of being hard-coded. An example implementation of how to accomplish this via Javascript is shown in Appendix C.

Step 3 validates that no changes were caused by conforming the tool's archiving procedure to the specification. The result matches that of Figure 33a, which shows that the archiving procedure again created an archive (*ArchiveFromAdapted*) of the user's profile on test.socialstandard.org.

Step 4 required manual verification with no deviation on the HTML or binary data preserved outside of timestamps generated for each respective session. The verification process showed that the results matched.

Step 5 requires the target website to be modified to simulate the event where

a social media website implements a hierarchical change. The synthetic website, created for this thesis, uses Apache .htaccess directives to prettify URIs to all direct to a single script handler. The directive originally used to remap http://test.socialstandard.org/personal is:

```
1 RewriteRule ^personal$ index.php?section=personal [NC]
```

Simply changing the redirect condition is sufficient to produce an HTTP 404 error when the aforemented URI is accessed.

```
1 RewriteRule ^myfeed$ index.php?section=personal [NC]
```

Step 6 consists of running the archive procedure again and Step 7 of verifying that making this change causes the tool to break. This behavior was common to the version of Archive Facebook that did not reference the specification and instead relied on a set of URIs to archive. Figure 34 shows that the output obtained in all of the previous steps is no longer achievable.

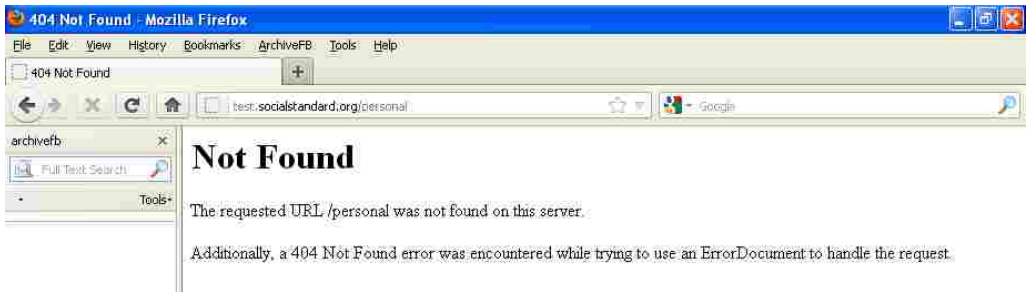


Fig. 34. The target website's URI scheme has changed. The new URI for the content that was previously at http://test.socialstandard.org/personal is now at http://test.socialstandard.org/myfeed.

Per Step 8, the remote specification is modified to reflect this change in the website's hierarchy. The change needed is shown in Figure 35, lines 3 and 4.

```
1     ...
2     <socialMediaWebsiteSection type="
          SocialMediaWebsiteSectionPersonalStream">
3       <name>Personal Stream</name>
4       <url>http://test.socialstandard.org/personal</url>
5       <preprocessor type="SocialMediaScrollPreprocessor">
6         <timeBetweenFirings>0</timeBetweenFirings>
7         <maxFirings>0</maxFirings>
8         <conditionBeforeSubsequentFiring></
              conditionBeforeSubsequentFiring>
9       </preprocessor>
10    </socialMediaWebsiteSection>
11    ...
```

(a)

```
1     ...
2     <socialMediaWebsiteSection type="
          SocialMediaWebsiteSectionPersonalStream">
3       <name>My Stream</name>
4       <url>http://test.socialstandard.org/myfeed</url>
5       <preprocessor type="SocialMediaScrollPreprocessor">
6         <timeBetweenFirings>0</timeBetweenFirings>
7         <maxFirings>0</maxFirings>
8         <conditionBeforeSubsequentFiring></
              conditionBeforeSubsequentFiring>
9       </preprocessor>
10    </socialMediaWebsiteSection>
11    ...
```

(b)

Fig. 35. A subtle change (lines 3 and 4) was made (from Figure 35a to Figure 35b) to the synthetic website's specification to change the location of the user's personal stream/feed as well as the name of the resource at the new location.

Proceeding with the last two steps results in the same screenshot as in Figure 36a.

(a)

(b)

Fig. 36. After conforming to the specification, the modified version of Archive Facebook is able to fetch and preserve any arbitrary collection of URI and associate them with one another through URIs rewriting. The end-result is a local navigable version of the specified website. Figure 36a shows the synthetic website at test.socialstandard.org has been preserved. Note the URI (annotated in 36a, shown more clearly in 36b) implicitly stored the date and time of archiving through the name of the directory created on the local machine.

VII.7 SUMMARY

In Chapter VII, an evaluation procedure is performed on aspects pertaining to this thesis using a synthetic social media website for which the hierarchy could be manipulated for the sake of testing. In Section VII.1, the synthetic site is described and the experiment setup. Section VII.2 proposed a hypothesis that the experiment is to validate. Section VII.3 discussed the applicable tool used for the experimentation. Section VII.4 formalized the experimental procedure to be run. Section VII.5 discussed in detail the changes made to the tool to make it more general purpose and more effective at accomplishing its task. Section VII.6 executed the experimental

procedure and expressed the results.

CHAPTER VIII

CONCLUSION AND FUTURE WORK

Content on the web residing behind authentication is not currently archived in a way that makes it accessible to tools like the Internet Archive's Wayback Machine. Social media websites require authentication, therefore, social media websites are not currently being archived in the same manner as the surface web. Tools that attempt to backup and archive social media content frequently do so in a manner that produces non-standard output. This makes them prone to breaking when the target websites' design or structure evolves.

The contributions of this thesis are as follows:

- Highlight difficulties in personal web archiving that until now have not been addressed.
- Recognize that many social media websites contain some degree of commonality in their respective hierarchical navigation schemes.
- Propose a way (Chapter V) to resolve the primary difficulty of personal web archiving tool breaking through the utilization of a remote specification.
- Evaluate the effectiveness of the specification through implementing conformance into an existing tool.
- Provide a reference implementation (WARCreate) for getting content behind the walled garden of authentication into a form (WARC) recognized as a standard for archiving content on the surface web.
- Leverage a client-side server suite (XAMPP) to execute scripts, normally requiring a separate server, to support personal web preservation initiated from a browser.

This thesis proposes a way to resolve the problem of archiving tools frequently breaking via the abstract specification and site-specific instantiation of the structure of these websites. By having tools use this specification as a source for what to archive

and how to do so (e.g., with the necessary preprocessing), the implementation of these tools will become more robust and standardized.

To demonstrate the robustness of the specification, I simulated a change (Chapter VII) to a synthetic website's hierarchy. The tools are shown to no longer function though they conform to the website-specific instance of the specification. The specification is then changed and the tool is shown to function again without the need to modify the tool's underlying code.

To overcome the problem of the preserved content not being in a standard portable format, I developed a tool (WARCreate) to convert any webpage, including those behind authentication, into a format consumable by wayback. This tool, too, was made to conform to the specification via the integration of procedures to sequentially archive pages (Section VI.2.1) based on the website currently being viewed. The success of this tool and the specification can be shown in the adaptability of a tool's code to the change of a target website (as previously mentioned) and the ability to replay content archived by WARCreate in wayback, respectively. The latter, though not the primary intent of this thesis, has a great potential for expansion, as the ability to archive this content in this way did not exist prior to the developments of this work.

Beyond WARCreate, this thesis also resolves other issues in personal web archiving (e.g., bringing wayback to the masses through the easy installer in Section IV.3) and validates its processes through integration with secondary technologies (e.g., Memento) beyond the initial scope of this research. The primary objective of this research was to resolve the issue personal web archiving tools have in ceasing to function when the target changes. By conforming tools to the specification, this problem is mitigated and in some cases, resolved.

Outstanding work remaining beyond the initial scope of this thesis is as follows:

- Expand the applicability of the specification to other social media websites
- Account for websites that do not follow a good accessibility model (e.g., sites that do not provide a unique URI for each section)
- Mature the development of WARCreate by leveraging the full WARC standard, directly implementing the wayback WARC library, improving user interface, etc.

- Account for more preprocessing actions in the specification
- Address facets of personal web archiving relating to perspective that were explored in Sections II.1, III.1 and III.2.

REFERENCES

[1] "PKCS #1 v2.3: RSA Cryptography Standard," RSA Laboratories, Tech. Rep., June 2002. [Online]. Available: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf

[2] "The web robots pages," 2007. [Online]. Available: http://www.robotstxt.org

[3] "Class AggressiveURLCanonicalizer," Internet Archive, Tech. Rep., Jan 2011, http://archive-access.sourceforge.net/projects/wayback/apidocs/index.html?org/archive/wayback/util/url/AggressiveUrlCanonicalizer.html.

[4] "DOMCrypt API Spec," Mozilla Foundation, Septmber 2011, https://wiki.mozilla.org/Privacy/Features/DOMCryptAPISpec/Latest.

[5] "Wayback Administrator Manual," Jan 2011, http://archive-access.sourceforge.net/projects/wayback/administrator_manual.html.

[6] "Wget with WARC Output," Archive Team, 2012, http://www.archiveteam.org/index.php?title=Wget_with_WARC_output.

[7] S. Ainsworth, A. AlSum, H. SalahEldeen, M. Weigle, and M. Nelson, "How much of the Web is Archived," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, vol. 11, 2011.

[8] M. K. Bergman, "The Deep Web: Surfacing Hidden Value," *Journal of Electronic Publishing*, vol. 7, no. 1, 2001.

[9] T. Berners-Lee, L. Masinter, and M. Mccahill, "RFC 1738: Uniform Resource Locator (URL)," http://www.ietf.org/rfc/rfc1738.txt.

[10] J. Bormans and K. Hill, "MPEG-21 Overview v.5," Organisation Internationale De Normalization, October 2002. [Online]. Available: http://mpeg.chiariglione.org/standards/mpeg-21/mpeg-21.htm

[11] R. G. Capra, C. A. Lee, G. Marchionini, T. Russell, C. Shah, and F. Stutzman, "Selection and Context Scoping for Digital Video Collections: An Investigation of YouTube and Blogs," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2008, pp. 211–220.

[12] K. Carpenter Negulsecu, "The Internet Archive - an Unorthodox Digital Repository Strategy," in *Future Perfect 2012*, March 2012. [Online]. Available: http://youtu.be/M9xifl3Ppnk

[13] C. Castillo, "Effective Web Crawling," *SIGIR Forum*, vol. 39, no. 1, pp. 55–56, Jun. 2005.

[14] S. Chakrabarti, M. Van den Berg, and B. Dom, "Focused Crawling: a New Approach to Topic-Specific Web Resource Discovery," vol. 31, no. 11. Elsevier, 1999, pp. 1623–1640.

[15] J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling Through URL Ordering," *Computer Networks and ISDN Systems*, vol. 30, pp. 161–172, 1998.

[16] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler," in *Proceedings of the Twenty-sixth International Conference on Very Large Databases*, 2000, pp. 200–209.

[17] "Reference Model for an Open Archival Information System (OAIS)," Consultative Committee for Space Data Systems, January 2002. [Online]. Available: http://www.library.cornell.edu/dlit/MathArc/web/resources/OAISReferenceModel--Jan2002--CCSDS650.0-B-1.pdf

[18] E. Crook, "Web Archiving in a Web 2.0 World," *Electronic Library, The*, vol. 27, no. 5, pp. 831–836, 2009.

[19] M. Dougherty, E. Meyer, C. Madsen, C. Van den Heuvel, A. Thomas, and S. Wyatt, "Researcher Engagement with Web Archives: State of the Art," August 2010. [Online]. Available: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1714997

[20] P. Eckersley, "How Unique Is Your Web Browser?" in *Privacy Enhancing Technologies*, vol. 6205. Springer, 2010, pp. 1–18.

[21] "Statement of Rights and Responsibilities," Facebook, Mar 2012. [Online]. Available: http://www.facebook.com/legal/terms

[22] "Introduction to Fedora Object XML (FOXML)," Fedora Project, Jan 2005. [Online]. Available: http://fedora-commons.org/download/2.0/userdocs/digitalobjects/introFOXML.html

[23] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A Large-Scale Study of the Evolution of Web Pages," in *In Procoeedings of the Twelfth WWW Conference*, 2003, pp. 669–678.

[24] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Addison-Wesley Professional, 1995.

[25] Gomita, "Scrapbook Firefox extension," 2011, http://amb.vis.ne.jp/mozilla/scrapbook/?lang=en.

[26] J. Gunderson, "W3C User Agent Accessibility Guidelines 1.0 for Graphical Web Browsers," *Universal Access in the Information Society*, vol. 3, no. 1, pp. 38–47, 2004.

[27] E. Hammer-Lahav, "The OAuth 1.0 Protocol, Internet RFC-5849," April 2010.

[28] G. Hodge, "Best Practices for Digital Archiving: An Information Life Cycle Approach," *D-Lib Magazine*, vol. 6, no. 1, 2000.

[29] Internet Archive - About IA. Internet Archive. [Online]. Available: http://www.archive.org/about/about.php

[30] B. Kahle, "Preserving the Internet," *Scientific American*, vol. 276, no. 3, pp. 82–83, Mar. 1997.

[31] M. Kelly and M. C. Weigle, "WARCreate - Create Wayback-Consumable WARC Files from Any Webpage," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, Washington, DC, June 2012, pp. 437–438.

[32] M. Klein and M. L. Nelson, "Evaluating Methods to Rediscover Missing Web Pages from the Web Infrastructure," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2010, pp. 59–68.

[33] M. Klein, M. Aly, and M. L. Nelson, "Synchronicity: Automatically Rediscover Missing Web pages in Real Time," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2011, pp. 475–476. [Online]. Available: http://doi.acm.org/10.1145/1998076.1998193

[34] M. Klein, J. L. Shipman, and M. L. Nelson, "Is This a Good Title?" in *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, 2010, pp. 3–12.

[35] J. A. Kunze, A. Arvidson, G. Mohr, and M. Stack, "WARC file format," Tech. Rep. ISO 28500:2009, 2009.

[36] S. Lawrence and C. Giles, "Searching the World Wide Web," *Science*, vol. 280, no. 5360, pp. 98–100, 1998.

[37] "Digital Natives Explore Digital Preservation," Library of Congress, 2010. [Online]. Available: http://www.digitalpreservation.gov/multimedia/videos/students10.html

[38] "Metadata Encoding and Transmission Standard (METS) Official Website," Library of Congress, June 2012. [Online]. Available: http://www.loc.gov/standards/mets/

[39] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker, "The LOCKSS Peer-to-Peer Digital Preservation System," *ACM Transactions on Computer Systems*, vol. 23, no. 1, pp. 2–50, 2005.

[40] C. C. Marshall, "Rethinking Personal Digital Archiving, Part 1: Four Challenges from the Field," *D-Lib Magazine*, vol. 14, no. 3/4, p. 2, 2008.

[41] ——, "Rethinking Personal Digital Archiving, Part 2: Implications for Services, Applications, and Institutions," *D-Lib Magazine*, vol. 14, no. 3/4, p. 2, 2008.

[42] ——, "Challenges and Opportunities for Personal Digital Archiving." Society of American Archivists, Chicago, IL, 2011, pp. 90–114.

[43] ——, "Ownership, Aggregation and Re-use of Personal Data." Presented at the 3rd Annual Conference on Personal Digital Archiving, San Francisco, CA, 2012. [Online]. Available: http://ia600807.us.archive.org/15/items/personaldigitalarchiving2012pt2/pda2012-17cathymarshall.ogv

[44] C. C. Marshall, S. A. Bly, and F. Brun-Cottan, "The Long Term Fate of Our Digital Belongings: Toward a Service Model for Personal Archives," in *Proceedings of IS&T Archiving 2006*, May 2006. [Online]. Available: http://www.csdl.tamu.edu/~marshall/archiving2006-marshall.pdf

[45] C. C. Marshall, F. McCown, and M. L. Nelson, "Evaluating Personal Archiving Strategies for Internet-based Information," in *Proceedings of IS&T Archiving 2007*, May 2007, pp. 151–156, (Also available as arXiv:0704.3647v1).

[46] C. C. Marshall and F. M. Shipman, "Attitudes About Institutional Archiving of Social Media." Presented at Archiving 2011, Salt Lake City, Utah, 2011. [Online]. Available: http://www.csdl.tamu.edu/~marshall/Archiving2011-Marshall-Shipman.pdf

[47] ——, "The Ownership and Reuse of Visual Media," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2011, pp. 13–17.

[48] J. Masanès, "Web Archiving: Issues and Methods," *Web Archiving*, pp. 1–53, 2006.

[49] Matasano Security, "Javascript Cryptography Considered Harmful," 2010, http://www.matasano.com/articles/javascript-cryptography/.

[50] F. McCown and M. Nelson, "A Framework for Describing Web Repositories," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2009, pp. 341–344.

[51] ——, "What Happens When Facebook is Gone?" in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2009, pp. 251–254.

[52] F. McCown, C. C. Marshall, and M. L. Nelson, "Why Web Sites Are Lost (and How They're Sometimes Found)," *Communications of the ACM*, vol. 52, no. 11, pp. 141–145, 2009.

[53] F. McCown and M. L. Nelson, "Agreeing to Disagree: Search Engines and Their Public Interfaces," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, New York, NY, USA, 2007, pp. 309–318.

[54] F. McCown, J. A. Smith, M. L. Nelson, and J. Bollen, "Lazy Preservation: Reconstructing Websites by Crawling the Crawlers," in *WIDM '06: Proceedings of the 8th Annual ACM International Workshop on Web Information and Data Management*, 2006, pp. 67–74.

[55] J. McHugh, "Should Web Giants Let Startups Use the Information They Have About You," *Wired Magazine (16.01)*, vol. 20, 2007.

[56] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic, "Introduction to Heritrix, an Archival Quality Web Crawler," in *4th International Web Archiving Workshop (IWAW04)*, Sep. 2004.

[57] C. Northern. (2009, September) Announcing ArchiveFacebook - A Firefox Add-on for Archiving Facebook Accounts. [Online]. Available: http://ws-dl.blogspot.com/2009/09/archivefacebook.html

[58] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web," in *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001, pp. 129–138. [Online]. Available: citeseer.ist.psu.edu/raghavan01crawling.html

[59] A. Ranganathan, "File API," W3C Working Draft, Nov 2009, http://www.w3.org/TR/2009/WD-FileAPI-20091117/.

[60] D. Recordon and D. Hardt, "The OAuth 2.0 Authorization Framework, Internet Draft," May 2012.

[61] D. Rosenthal, "Harvesting and Preserving the Future Web," http://blog.dshr.org/2012/05/harvesting-and-preserving-future-web.html.

[62] ——, "JCDL 2010 Keynote," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, June 2010. [Online]. Available: http://blog.dshr.org/2010/06/jcdl-2010-keynote.html

[63] J. Rothenberg, "Avoiding Technological Quicksand - Finding a Viable Technical Foundation for Digital Preservation," Council on Library and Information Resources, Tech. Rep., January 1999.

[64] M. B. Saad and S. Gançarski, "Archiving the Web Using Page Changes Patterns: A Case Study," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2011, pp. 113–122.

[65] M. Smith, "Eternal Bits [Digital Files Preservation]," *IEEE Spectrum*, vol. 42, no. 7, pp. 22–27, 2005.

[66] M. Spaniol, A. Mazeika, D. Denev, and G. Weikum, "'Catch Me If You Can': Visual Analysis of Coherence Defects in Web Archiving," in *Proceedings of the 9th International Web Archiving Workshop (IWAW)*, 2009, pp. 27–37.

[67] S. Strodl, P. Beran, and A. Rauber, "Migrating Content in WARC Files," in *Proceedings of the 9th International Web Archiving Workshop (IWAW)*, 2009, pp. 43–49.

[68] A. Thomas, E. Meyer, M. Dougherty, C. Van den Heuvel, C. Madsen, and S. Wyatt, "Researcher Engagement with Web Archives: Challenges and Opportunities for Investment," Final Report for the JISC-funded project 'Researcher Engagement with Web Archives', Tech. Rep., 2010. [Online]. Available: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1715000

[69] H. Van de Sompel, M. L. Nelson, R. Sanderson, L. L. Balakireva, S. Ainsworth, and H. Shankar, "Memento: Time Travel for the Web," Tech. Rep. arXiv:0911.1112, 2009. [Online]. Available: http://arxiv.org/pdf/0911.1112

[70] P. Windrum, "Leveraging Technological Externalities in Complex Technologies: Microsoft's Exploitation of Standards in the Browser Wars," *Research Policy*, vol. 33, no. 3, pp. 385–394, 2004.

[71] A. Wright, "Exploring a 'Deep Web' That Google Can't Grasp," *New York Times*, vol. 23, p. B1, Feburary 22, 2009.

APPENDIX A

SPECIFICATION XML FOR FACEBOOK

The contents of this appendix consist of an XML representation of the content breakup of Facebook as of early 2012. An updated version of this definition is available at http://spec.socialstandard.org/facebook.xml. Reference to other instantiations can be found at http://spec.socialstandard.org.

```
1  <?xml version="1.0" ?>
2  <?xml-stylesheet type="text/xsl" href="socialStandard.xslt" ?>
3  <socialMediaWebsite>
4    <homepage>http://www.facebook.com</homepage>
5    <sections>
6      <socialMediaWebsiteSection type="
           SocialMediaWebsiteSectionPersonalStream">
7        <name>Wall</name>
8        <url>http://www.facebook.com/profile.php?sk=wall</url>
9        <preprocessor type="SocialMediaScrollPreprocessor">
10         <timeBetweenFirings>0</timeBetweenFirings>
11         <maxFirings>0</maxFirings>
12         <conditionBeforeSubsequentFiring></
               conditionBeforeSubsequentFiring>
13       </preprocessor>
14     </socialMediaWebsiteSection>
15     <socialMediaWebsiteSection type="
           SocialMediaWebsiteSectionUserInfo">
16       <name>Info</name>
17       <url>http://www.facebook.com/profile.php?sk=info</url>
18     </socialMediaWebsiteSection>
19     <socialMediaWebsiteSection type="
           SocialMediaWebsiteSectionMultimediaCollection">
20       <name>Photos</name>
21       <url>http://www.facebook.com/profile.php?sk=photos</url>
22       <preprocessor type="SocialMediaScrollPreprocessor">
23         <timeBetweenFirings>0</timeBetweenFirings>
24         <maxFirings>0</maxFirings>
25         <conditionBeforeSubsequentFiring></
               conditionBeforeSubsequentFiring>
```

```
26        </preprocessor>
27      </socialMediaWebsiteSection>
28      <socialMediaWebsiteSection type="
            SocialMediaWebsiteSectionMultimediaCollection">
29        <name>Notes</name>
30        <url>http://www.facebook.com/profile.php?sk=notes</url>
31      </socialMediaWebsiteSection>
32      <socialMediaWebsiteSection type="SocialMediaWebsiteFriends">
33        <name>Friends</name>
34        <url>http://www.facebook.com/profile.php?sk=friends</url>
35        <preprocessor type="SocialMediaScrollPreprocessor">
36          <timeBetweenFirings>0</timeBetweenFirings>
37          <maxFirings>0</maxFirings>
38          <conditionBeforeSubsequentFiring></
                conditionBeforeSubsequentFiring>
39        </preprocessor>
40      </socialMediaWebsiteSection>
41    </sections>
42  </socialMediaWebsite>
```

APPENDIX B

TABULAR COMPARISON OF TOOLS IN EVALUATION

Users currently have access to a wide array of tools to accomplish the task of preserving their personal data. Listed here is a high-level comparison of these tools in tabular form.

	website agnostic	automated sequential archiving	preserves look-and-feel	content comprehensive (no exclusions)	An archive rather than a backup	Browser-based Execution	Browser-based replay
Facebook Data Dump	✗	N/A	✗	✗*(agency)*	✗	✗*(server)*	✓
Google Takeout	✗	N/A	✗	✗*(agency)*	✗	✗*(server)*	✓
WARCreate	✓	✗	✓*(Chrome)*	✓	✓	✓	✗
Archive Facebook	✗	✓	✓*Firefox*	✓	✗	✓	✓*(Firefox)*
"Save Webpage As"	✓	✗	✓	✓	✗	✓	✓
OpenSocial	✗	N/A					
wget with authentication	✓	✗	✗ *wget user agent*	✓	✗	✗	✗
wget-warc	✓	✗	✗ *wget user agent*	✓	✓	✗	✗

APPENDIX C

CODE TO CAPTURE ANY SPEC-DEFINED SITE

```
ssCapture : function(specIn) {
    var parser = new DOMParser();
    var xml = parser.parseFromString(specIn, "text/xml");

    function SocialMediaWebsiteSection(asElement){
      this.element = asElement;
      var urls = this.element.getElementsByTagName("url");

      this.url = $(urls[0]).text(); //potentially a regex
      var childrenElements = this.element.getElementsByTagName("
          children");
      var childrenElement, names;
      if(childrenElements.length > 0){
        childrenElement = childrenElements[0];
        names = childrenElement.getElementsByTagName('name');
        this.childName = $(names[0]).text();
        this.subsections = [];
      }
    }

    SocialMediaWebsiteSection.prototype.isARegEx = function (){
      return (this.url.indexOf("[")   != -1
           || this.url.indexOf("(")   != -1
           || this.url.indexOf("%5B") != -1
           || this.url.indexOf("%5D") != -1);
    };

    SocialMediaWebsiteSection.prototype.addSubSection = function() {
      for(var i=0; i<subsections.length; i++){
        if(!(this.subsections[i])){ //associate child with parent

        }
      }
    };
```

```
34
35      var sectionsElements = xml.getElementsByTagName("
            socialMediaWebsiteSection");
36      var sections = [];
37      for(var i=0; i<sectionsElements.length; i++){
38        var potentialSection = new SocialMediaWebsiteSection(
              sectionsElements[i]);
39        var obj;
40        if(potentialSection.isARegEx()){sections.push(potentialSection)
              ;
41        }else {
42          obj = potentialSection;
43          sections.push(obj);
44        }
45      }
```

APPENDIX D

SAMPLE WARC FILE

```
1  WARC/1.0
2  WARC-Type: warcinfo
3  WARC-Date: 2012-07-25T02:51:27.573Z
4  WARC-Filename: 2fd0e5b61c911f11c167ddec14320a73.warc
5  WARC-Record-ID: <urn:uuid:b53124ec-7496-1438-d4d1-74c3e32b552e>
6  Content-Type: application/warc-fields
7  Content-Length: 483
8
9  software: WARCreate/0.2012.7.23 http://matkelly.com/warcreate
10 format: WARC File Format 1.0
11 conformsTo: http://bibnum.bnf.fr/WARC/
       WARC_ISO_28500_version1_latestdraft.pdf
12 description: recurrence=ANNUAL, maxDuration=432000, maxDocumentCount
       =1000000, isTestCrawl=false, seedCount=61, accountId=89
13 robots: classic
14 http-header-user-agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit
       /536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11
15 http-header-from: warcreate@matkelly.com
16
17
18 WARC/1.0
19 WARC-Type: request
20 WARC-Target-URI: https://twitter.com/#!/search/#digpres12
21 WARC-Date: 2012-07-25T02:51:27.573Z
22 WARC-Concurrent-To: <urn:uuid:9cb4670a-98cc-8748-d93a-9b55f53cfb0d>
23 WARC-Record-ID: <urn:uuid:8cdcac72-5518-6b1d-e9f4-c12ded15a0e9>
24 Content-Type: application/http; msgtype=request
25 Content-Length: 327
26
27
28 GET /#!/search/#digpres12 HTTP/1.1
29 Host: twitter.com
30 Connection: close
31 User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.11 (KHTML,
       like Gecko) Chrome/20.0.1132.57 Safari/536.11
```

```
Accept-Encoding: gzip
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7
Cache-Control: no-cache
Accept-Language: de,en;q=0.7,en-us;q=0.3

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: https://twitter.com/#!/search/#digpres12
WARC-Date: 2012-07-25T02:51:27.573Z
WARC-Concurrent-To: <urn:uuid:dddc4ba2-c1e1-459b-8d0d-a98a20b87e96>
WARC-Record-ID: <urn:uuid:6fef2a49-a9ba-4b40-9f4a-5ca5db1fd5c6>
Content-Type: application/warc-fields
Content-Length: 49

outlink: https://twitter.com/#!/search/#digpres12

WARC/1.0
WARC-Type: response
WARC-Target-URI: https://twitter.com/#!/search/#digpres12
WARC-Date: 2012-07-25T02:51:27.573Z
WARC-Record-ID: <urn:uuid:83431648-8236-7d75-f8aa-fd1c371dfe09>
Content-Type: application/http; msgtype=response
Content-Length: 142997

HTTP/1.1 200 OK
Content-Type: text/html
Date: Tue Jul 24 2012 22:51:27 GMT-0400 (Eastern Daylight Time) GMT
Last-Modified: Tue Jul 24 2012 22:51:27 GMT-0400 (Eastern Daylight
    Time) GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
Accept-Ranges: bytes
Content-Type: text/html

<html class="   js"><head>

    <title>Twitter / Search - #digpres12</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta charset="utf-8">

    (content removed for ease of viewing)

```

```
72
73
74 </body></html>
75
76
77 WARC/1.0
78 WARC-Type: request
79 WARC-Target-URI: https://twimg0-a.akamaihd.net/a/1343165977/t1/css/
       t1_more.bundle.css
80 WARC-Date: 2012-07-25T02:51:27.573Z
81 WARC-Concurrent-To: <urn:uuid:9cb4670a-98cc-8748-d93a-9b55f53cfb0d>
82 WARC-Record-ID: <urn:uuid:39a45298-ea73-7103-8afe-62876587a0f2>
83 Content-Type: application/http; msgtype=request
84 Content-Length: 370
85
86
87 GET /a.akamaihd.net/a/1343165977/t1/css/t1_more.bundle.css HTTP/1.1
88 Host: twimg0-a.akamaihd.net
89 Connection: close
90 User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.11 (KHTML,
       like Gecko) Chrome/20.0.1132.57 Safari/536.11
91 Accept-Encoding: gzip
92 Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7
93 Cache-Control: no-cache
94 Accept-Language: de,en;q=0.7,en-us;q=0.3
95
96 WARC/1.0
97 WARC-Type: response
98 WARC-Target-URI: https://twimg0-a.akamaihd.net/a/1343165977/t1/css/
       t1_more.bundle.css
99 WARC-Date: 2012-07-25T02:51:27.573Z
100 WARC-Record-ID: <urn:uuid:ce07dfc4-7b08-c8c1-b310-f55d3529a443>
101 Content-Type: application/http; msgtype=response
102 Content-Length: 125861
103
104 HTTP/1.1 200 OK
105 Content-Type: text/css
106 Date: Tue Jul 24 2012 22:51:27 GMT-0400 (Eastern Daylight Time) GMT
107 Last-Modified: Tue Jul 24 2012 22:51:27 GMT-0400 (Eastern Daylight
       Time) GMT
108 Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
```

```
109 Accept-Ranges: bytes
110 Content-Type: text/css
111
112 .btn{position:relative;display:inline-block;overflow:visible;padding
       :5px 10px;font-size:13px;font-weight:bold;line-height:18px;color
       :#333;text-shadow:0 1px 0 rgba(255,255,255,.5);background-color:#
       ccc;background-repeat:no-repeat;border:1px solid #ccc;cursor:
       pointer;-webkit-border-radius:4px;-moz-border-radius:4px;border-
       radius:4px;border-radius:0 \0;-webkit-box-shadow:0 1px 0 rgba
       (255,255,255,.5);-moz-box-shadow:0 1px 0 rgba(255,255,255,.5);box
       -shadow:0 1px 0 rgba(255,255,255,.5);}
113
114 (content removed for ease of viewing)
```

VITA

Matthew Ryan Kelly
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

EDUCATION

M.S. in Computer Science, Old Dominion University, 2012
B.S. in Computer Science, University of Florida, 2006

EMPLOYMENT

6/11 to Present	Mobile Applications Developer/Programmer for NASA Langley Research Center via Science Systems and Applications, Inc. (Hampton, Virginia)
5/11 to Present	Research Assistant for Old Dominion University (Norfolk, Virginia)
8/09 to 12/09	Research Intern for BMW Group (Greenville, South Carolina)
4/06 to Present	Web Programmer for Blade Agency (Gainesville, Florida)
4/06 to 2/07	Web Developer for & MindSolve Technologies/SumTotal Systems, Inc. (Gainesville, Florida)
5/05 to 4/06	Web Designer for Center for Instructional Technology and Training at University of Florida (Gainesville, Florida)
7/04 to 1/06	Managing Editor, New Media at The Independent Florida Alligator (Gainesville, Florida)
7/04 to 5/05	Web Designer for Office of Student Activities at University of Florida (Gainesville, Florida)

PUBLICATIONS AND PRESENTATIONS

A complete list is available at http://matkelly.com/pubs

PROFESSIONAL AFFILIATIONS

Association for Computing Machinery (ACM)

Permalinks

Homepage: http://matkelly.com
Email: me@matkelly.com

Typeset using LaTeX.