LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

**Webarchiving / webcrawling**

LIMA is the international platform for sustainable access to media art in the Netherlands. LIMA offers digitisation services and advice to museums, artists, and private collectors and has a cross-institutional, domain-specific, digital repository for digital art, where media artworks from over 30 media art collections in the Netherlands are preserved for the future. LIMA has implemented maintenance procedures for computer-based artworks, such as testing equipment, producing backups, documenting software, storage, and checking in its workflow. And an operational repository for digital art. We are currently updating our digital repository and workflow. With an emphasis on net art and complex digital artworks, our new dynamic repository, collection information system, and associated workflows, should be suited to capturing the mutability inherent throughout the lifecycle of such works. In this scope webarchiving /webcrawling was researched.

**The WARC File Standard**

With the (partial) exception of one, all of the methods discussed herein make use of the WARC file standard[1], which makes some background information of use. WARC is built upon the ARC file format previously commonly used to store content blocks harvested from the World Wide Web. Prompted by the International Internet Preservation Consortium, the aim of the WARC standard is to provide a standardized format way to structure, manage, and store resources harvested from the web. It has been standardized, and made public as ISO28500:2009.

The structure of the WARC archive consists of content blocks, bound in header data. Hence a WARC file begins with a WARCINFO block describing the content of subsequent records (filename, content type, record identifiers etc.), followed

1 https://www.loc.gov/preservation/digital/formats/fdd/fdd000236.shtml

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

by protocol responses (target URL/IP address, time stamps) and the means by which content was requested (i.e. http GET commands). Subsequent to this is the content data as crawled/harvested.

From the perspective of preservation best practice, it is laudable that an open and publicly defined standard has been developed, though it must be noted that the WARC standard is not completely prescriptive, representing instead somewhat of a loose standard – data can be pulled, encoded, and stored in a number of ways whilst remaining within the bounds of the standard. For example, some WARC tools automatically compress the downloaded data, some leave it human-readable as far as is possible (i.e. in the case of non-binary content). Similarly, some engines request/write related metadata to a different degree. This can be shown by presenting the warcinfo block created by Heritrix (with default settings) and Webrecorder respectively, when pointed at the same set of Constant Dullaart websites:

WARC/1.0
WARC-Type: warcinfo
WARC-Date: 2018-04-11T11:19:34Z
WARC-Filename:
MAT-20180411111934190-00000-1107~10.10.0.74~8443.warc
WARC-Record-ID: <urn:uuid:1cfbf7ab-b560-4a14-bc0c-4a802a388ae7>
Content-Type: application/warc-fields
Content-Length: 380

software: Heritrix/3.2.0 http://crawler.archive.org
ip: 10.10.0.74
hostname: 10.10.0.74
format: WARC File Format 1.0
conformsTo:
http://bibnum.bnf.fr/WARC/WARC_ISO_28500_version1_latestdraft.pdf
isPartOf: basic

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

description: Basic crawl starting with useful defaults

robots: ignore

http-header-user-agent: Mozilla/5.0 (compatible; heritrix/3.2.0 +http://yourdomain.com)

WARC/1.0
WARC-Type: warcinfo
WARC-Record-ID: <urn:uuid:712786d8-3d90-11e8-88fd-0242ac130002>
WARC-Filename: constant-dullaart-20180411135841.warc.gz
WARC-Date: 2018-04-11T13:58:41Z
Content-Type: application/warc-fields
Content-Length: 243

software: Webrecorder Platform v3.15
format: WARC File Format 1.0
creator: jwraith
isPartOf: Constant%20Dullaart
json-metadata: {"title": "Constant Dullaart", "type": "collection", "size": 220163788, "desc": "", "created_at": 1519034097}

A tangential benefit of the use of an open standard is the prevailing development of an ecosystem of free/libre and open source softwares (and support softwares) for the creation, viewing, manipulation and management of WARC files[2]. As can be expected of FLOSS software, these span a wide range of utility value, functional coherence, and ease of use, though these efforts are of course commendable[3].

2 See https://www.archiveteam.org/index.php?title=The_WARC_Ecosystem for an extensive, if not exhaustive, list of WARC-related softwares as of March 22nd 2018.

3 It would seem of interest, given the task in hand, to

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

A software tool of paramount importance to the present work is OpenWayback[4], an open source iteration of the Internet Archive's Wayback Machine, that allows

for the collation and viewing of a number of different captures of a given website across different dates. Similarly, Web Archiving Integration Layer [WAIL] within which OpenWayback comes bundled, proved invaluable in mitigating a huge amount of the work required in installing and setting up Heritrix.

## WGET

*Wget* represents the most prosaic of the tools under scrutiny. *Wget* is a standard free software distributed (initially) as part of the GNU project (though since ported to a range of other systems since[5]), designed for the retrieval of files, using the HTTP(S) or FTP(S) protocols. As a relatively simple command-line tool, it is amenable to its being used as part of a script (for example, to undertaking the retrieval of a given website on a scheduled basis, as part of a *cron* job). As a simple, well-supported piece of software, *wget* proves robust, able to to work across slow/intermittent network connections, and can be set up to check timestamps of target material to ensure that only files newer than those stored in a local repository are downloaded[6]. By default, *wget* outputs 1:1 copies of the files it encounters

Whilst a number of front-ends for *wget* are extant[7], the underlying functionality remains the same, and is dependent on the user spending some time to understand the ways in which *wget* can be used. In the first instance, this entails using it in a "polite" manner, that does not cause undue strain on the target server(s). Further, this could involve knowing in advance whether a given crawling exercise should span across different domains, to what depth links should be followed (that is to say, if one is downloading "index.html" which itself

4 http://netpreserve.org/web-archiving/openwayback/
5 cf. http://gnuwin32.sourceforge.net/packages/wget.htm for a Windows version
6 For an idea of the range of adaptability/customizabilty, refer to the manual: https://www.gnu.org/software/wget/manual/wget.html
7 e.g. https://sites.google.com/site/visualwget/a-download-manager-gui-based-on-wget-for-windows

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

contains a link to “page2.html”, is page2.html to be downloaded? And if page2.html links to a page3.html, should page3.html be downloaded? and so on). It is this recursion that allows *wget* to function in a manner understandable

as a webcrawler, semi-autonomously navigating through given data and using what is contained therein to direct it further. There are some limitations to this that could potentially be pertinent to the archiving of net art: most notably, it only works to retrieve copies of remote files, without executing any of the code therein. This means that links contained in Javascript can be followed, but content generated on the execution of Javascript will not be. This extends to all kinds of interaction inherent in a given site: *wget* only returns the data that comprise the site and, in common with all softwares under review here, it does nothing to support the retention of network infrastructures that may play an important role in the presentation of a given site.

**webrecorder.io**

Webrecorder.io is a web preservation tool designed and presented by Rhizome. It proffers a very simple functionality – enter the URL of the site to be recorded, and then navigate through said site at one’s ease. At the time of writing a limited number of browser/browser versions are able to be emulated (Chrome v53 & v60, and Firefox v49, v56 & v57), running from within an emulated Linux environment[8]. The main strength of of webrecorder is its accessibility and ease of use: one can commence by entering a URL, and pressing the record button. Once the emulated environment is loaded, harvesting is directed by the user’s interaction with the site, downloading content as it is encountered. As such, it cannot be understood to crawl the content, as wget does, rather it is dependent on being directed. This provides the inspiration for the name, though this proves something of a misnomer: whilst the webrecorder records what the user follows,

[8] I say ‘pertained’ as no information is given as to the nature of this emulation, or which particular Linux distribution/version thereof is being emulated. A potentially interesting development of this functionality would be some further integration of the oldweb.today service, also produced by Rhizome, which presents a broader range of platforms/browsers, but is presently limited

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

it does not record the specificities of the interaction one might associate with the concept of 'recording'.

The resulting recordings can be organized into collections within the Webrecorder site, and said collections downloaded as a gzip compressed WARC file. During this process, all links are changed to refer to URLs internal to said WARC file. Whilst this is a reasonable step within an enclosed site, ensuring that all links refer to the version saved at that specific instance, it proves problematic with more diffuse works. In the example of the Constant Dullaart sites reliant on accessing the Google API through a proxy (cf. *The Revolving Internet*), it renders the site inoperable. Aside from this, my personal experience with webrecorder presented further (non-fatal) problems: the site proved dysfunctional when using Mozilla Firefox, requiring the use of Chrome. Further, naively using the URL of a site to be recorded as the title of the recording caused some confusion when played back, as the combination of the aforementioned changing of links *included* the name of the title in the new URL, thus embedding the URL of the original piece within links that point to somewhere within the archive. Whilst webrecorder serves as an exceptionally easy to use tool, these limitations make it sub-optimal in terms of it being any sort of "industry strength" archival/preservational tool. Similarly, the lack of options or customizability of one's activities within webrecorder serve to cast some doubt on the manner in which it performs its tasks[9].

**Heritrix**

Heritrix, developed by the Internet Archive, described itself as an "open-source, extensible, web-scale, archival-quality web crawler project". It is, broadly speaking, the industry standard as is illustrated by the institutions using it

[9] It should be noted that webrecorder is open-source, so anybody with the relevant understanding of coding should be able to find out, and change any of this should they desire. It perhaps goes without saying that this level of understanding is beyond that of your author…

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

(Koninklijke Bibliotheek, Internet Memory Foundation, Library of Congress, Smithsonian Institutes, Bibliothèque national de France et al.) Programmed in Java, and highly configurable, Heritrix aspires to cover as many types of

webcrawling as can be thought of, from the broad to the specific. Installation is dependent on the provision of a server, virtual or otherwise – to ease the rigours of doing this, I utilized a version bundled in an application called WAIL [Web Archiving Integration Layer][10]. This comes bundled with the Open Wayback engine for the viewing/playback of downloaded material, and is configured to provide access to both hosted on *localhost*.

The highly configurable nature of Heritrix is dependent on the correct setting up of the crawling task; this is dependent on some knowledge of coding, particularly JavaBeans, as exemplified by this example of a job setup:

```
 <bean id="longerOverrides"
class="org.springframework.beans.factory.config.PropertyOverrideConfigurer">
  <property name="properties">
   <props>
    <prop key="seeds.textSource.value">

# URLS HERE
http://jennifer.ps/
http://ooooooooooooooooooooooooooooooooooo.ooo/
http://ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
.ooooooooooooooooooooooooooooooooooooooo.ooo/
http://thedoubtinginternet.com/
http://thedisagreeinginternet.com/
http://wavingocean.com/
    </prop>
   </props>
```

[10] https://machawk1.github.io/wail/

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

```
  </property>
 </bean>



<!-- CRAWL METADATA: including identification of crawler/operator -->
 <bean id="metadata" class="org.archive.modules.CrawlMetadata"
autowire="byName">
      <property name="operatorContactUrl"
value="YOURCONTACTINFOHERE"/>
      <property name="jobName" value="MyWAILBasedHeritrixCrawl"/>
      <property name="description" value="SampleCrawl"/>
  <property name="robotsPolicyName" value="ignore"/>
  <!-- <property name="operator" value=""/> -->
  <!-- <property name="operatorFrom" value=""/> -->
  <!-- <property name="organization" value=""/> -->
  <!-- <property name="audience" value=""/> -->
   <property name="userAgentTemplate"
        value="Mozilla/5.0 (compatible; heritrix/@VERSION@
+@OPERATOR_CONTACT_URL@)"/>

 </bean>
```

In its default state, Heritrix outputs uncompressed WARC files, and institutes a similar “walled-garden” as discussed in the context of webrecorder’s attempts to navigate diffuse works. As the crawling process does not allow for any given input into the Google API presented by, say, *The Disagreeing Internet*, we are left with the vast range of the work’s potential states remaining unarchived. Indeed, unless one were to institute some Borges-like process of entering every possible combination of inputs, this remains unfeasible whilst links are replaced with those pointing inside the archive. A similar problem can be found in

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

attempting to crawl pages dependent on the use of scripting, whereby the engine will download, for example, Javascript code without executing it[11]; this potentially

leads to content being missed by the crawl. The crawler, in this regard, is something of a "dumb" tool: particularly in the case of works of net art, this obliviousness to user interactions/behaviours, is a not unimportant stumbling block. That said, of the options under study in the present work, Heritrix represents a stable, strong, and well-supported software with a broad base of users.

**Broader questions/conclusions**

In comparing the functions of these softwares as preservational tools, it is important to look beyond their own capabilities to what use they would be to an organization such as LIMA. Each presents problems in terms of the *fidelity* of reproduction that may or may not prove terminal to the preservation of a net art piece. Most webcrawling or webarchiving tools are not intended – though they may in practice do so, depending on the task at hand - to work with the level of exactitude and fidelity of representation required in maintaining the essential characteristics of net art. Rather, though this of course differs across the range of tools, the emphasis is often on the *breadth* of the harvesting.

So what one ends up with, in one format or another, is the code required to reconstruct or rehost a web-based work, with the caveat that some more involved work may be required than expected in order for a crawled version to "go live" (the nature of net art, of course, means that a site being preserved is dependent on it being active and accessible). As discussed previously, this can prove problematic in the case of diffuse works, whether they can be encapsulated in a WARC, and what sense is there in doing so?

[11] For a comprehensive discussion of this problem, see Brunelle, J., Weigle, M. C., & Nelson, M. L., "Archival Crawlers and JavaScript: Discover More Stuff but Crawl More Slowly", *ACM/IEEE Joint Conference on Digital Libraries*, June 2017, pp.1-10

LIMA
Arie Biemondstraat 111
1054 PD Amsterdam
The Netherlands
+31 (0)20 389 20 30
info@li-ma.nl | www.li-ma.nl

More broadly speaking, it is important in the context of net art to consider the importance of network infrastructures beyond mere content code: aside from, on the basic level, needing to understand the structure of a given piece when setting up a crawl job (knowing what depth to follow links, whether to span across different hosts etc.), more complicated setups can elude archiving. What is one

to make of the kind of dynamism found in jodi.org, where accessing said domain serves only to redirect the user to an (apparently randomly selected) site from a constellation of different sites ran by jodi? Given the hackerish nature of net art, and the frequent attempts to tinker with the basic structure of the internet on the part of net artists, such concerns are vital.

It is hard to envisage a situation where the use of web archiving software would be applicable as a preservation tool, given the aforementioned questions around fidelity and exactitude; more pragmatically, it would be expected that the vast majority of possible use cases would involve the provision of source code direct from an artist/institution, or representative thereof.

It could be argued that the retention of protocol message headers/server commands etc. in a log file could be of benefit in delineating the specific function of some websites, though it is imagined that a number of tools more suited to this task are in existence. Opening one's imagination slightly further, it might be argued that these softwares could be of use in some research capacity: regularly harvesting mutable sites over a period of time to assess their changes, or the use of mass-crawling to build corpora of net art works in order to complete some level of content or structural analysis.

*Jim Wraight*
*May 2018*