



International Digital Curation Conference
Bristol, UK, 5-7 December 2011

**Digital Forensics Formats:
seeking a storage format for web archiving**

Yunhyong Kim & Seamus Ross

Humanities Advanced Technology and Information Institute (HATII)
University of Glasgow, Glasgow, UK

&

Faculty of Information
University of Toronto, Toronto, Canada

yunhyong.kim@glasgow.ac.uk seamus.ross@utoronto.ca

This work is supported by ...

The BlogForever Project:

Co-funded by the European Union's Seventh Framework Programme (FP7-ICT-2009-6) under grant agreement n° 269963.

The BlogForever project (<http://www.blogforever.eu>) aims to establish best practices and an associated software platform capable of aggregating, preserving, managing and disseminating blogs.

Previous studies

Previous works have focused on the study of formats for isolated object types.

The storage of groups of objects have been handled by packaging formats that aggregate constituent object formats.

Objects found on the web:

- frequently reference other objects;
- represents dynamically changing information and ownership;
- harbour varying permission levels associated with their components.


The web is huge. The storage architecture selected must support scalability, in terms of:

- storage,
- seachability, and,
- its ability to support newly emerging complex functionalities (e.g. data intensive machine learning techniques).


Information producers on the web

WHERE CITATIONS COME FROM:


CITOGENESIS STEP #1:
THROUGH A CONVOLUTED PROCESS, A USER'S BRAIN GENERATES FACTS. THESE ARE TYPED INTO WIKIPEDIA.
THE "SCROLL LOCK" KEY WAS DESIGNED BY FUTURE ENERGY SECRETARY STEVEN CHU IN A COLLEGE PROJECT.



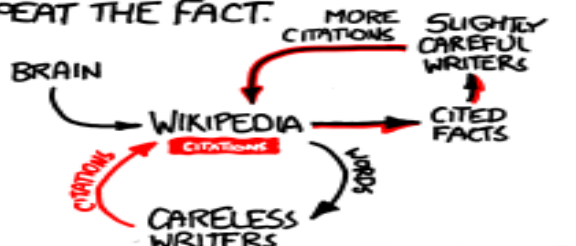
A RUSHED WRITER CHECKS WIKIPEDIA FOR A SUMMARY OF THEIR SUBJECT.
US ENERGY SECRETARY STEVEN CHU, (NOBEL PRIZEWINNER AND CREATOR OF THE UBIQUITOUS "SCROLL LOCK" KEY) TESTIFIED BEFORE CONGRESS TODAY...



SURPRISED READERS CHECK WIKIPEDIA, SEE THE CLAIM, AND FLAG IT FOR REVIEW. A PASSING EDITOR FINDS THE PIECE AND ADDS IT AS A CITATION.
GOOGLE IS YOUR FRIEND, PEOPLE.
<REF>{{CITE WEB|URL=



STEP #4
NOW THAT OTHER WRITERS HAVE A REAL SOURCE, THEY REPEAT THE FACT.



REFERENCES PROLIFERATE, COMPLETING THE CITOGENESIS PROCESS.

Permanent link to this comic: <http://xkcd.com/978/>

Image URL: <http://imgs.xkcd.com/comics/citogenesis.png>

This work is licensed under a Creative Commons Attribution-NonCommercial 2.5

License. This means you're free to copy and share these comics (but not to sell them).

In fact ...

In the digital environment, provenencial evidence surrounding digital objects can be derived from information external to the object such as:

- file modification dates and user name of the modifier,
- lists of files that were deleted,
- logs of processes (e.g. installation of programs) and resulting errors, and,
- trails of programs that had been run on the system.

This kind of history is retained on the system disk, as a result of often tacitly understood standard practice in software design and systems administration (e.g. see <http://c2.com/cgi/wiki?LoggingBestPractices>).

This is information that should be retained to trace accountability (not only with respect to humans but also software and hardware).

1. Formats for packing, storing, transferring, and backing-up the content (e.g. tar, International Internet Preservation Consortium WARC, AXF).
2. Formats that capture raw data, intended for recovery or installation (e.g. partimage, dd raw image).
3. Combination of formats listed in item 1 and 2) and standard compression methods (e.g. gzip, zip, bzip2, lzma).
4. Formats that combine packing and compression (e.g. 7-zip, PeaZip).
5. Forensic disk image formats (e.g. aff, aff4).

Criteria

Criteria	Related Library of Congress SF	Description
Completeness	N/A, only discussion re validation tools	access to full syntactic, semantic and pragmatic information
Recoverability	N/A, only discussion about not using compression and encryption	localised damage control
Validation	Disclosure	e.g. piecewise hashing
Scalability	N/A	Influence on process efficiency
Transparency	Disclosure, Impact for patents, Transparency	e.g. open to direct analysis with basic tools
Metadata flexibility	Self documentation	user defined metadata
Flexible data handling	External dependencies	restrictions on source, size, and type of platform

Library of Congress sustainability factors from:
<http://www.digitalpreservation.gov/formats/sustain/sustain.shtml>

Criteria

Criteria	Related Library of Congress SF	Description
Completeness	N/A, only discussion about available tools	access to full syntactic, semantic and pragmatic information
Recoverability	N/A, only discussion about not using compression and encryption	localised damage control
Validation	Disclosure	e.g. piecewise hashing
Scalability	N/A	Influence on process efficiency
Transparency	Disclosure, Impact for patents, Transparency	e.g. open to direct analysis with basic tools
Metadata flexibility	Self documentation	user defined metadata
Flexible data handling	External dependencies	restrictions on source, size, and type of platform

Library of Congress sustainability factors from:
<http://www.digitalpreservation.gov/formats/sustain/sustain.shtml>

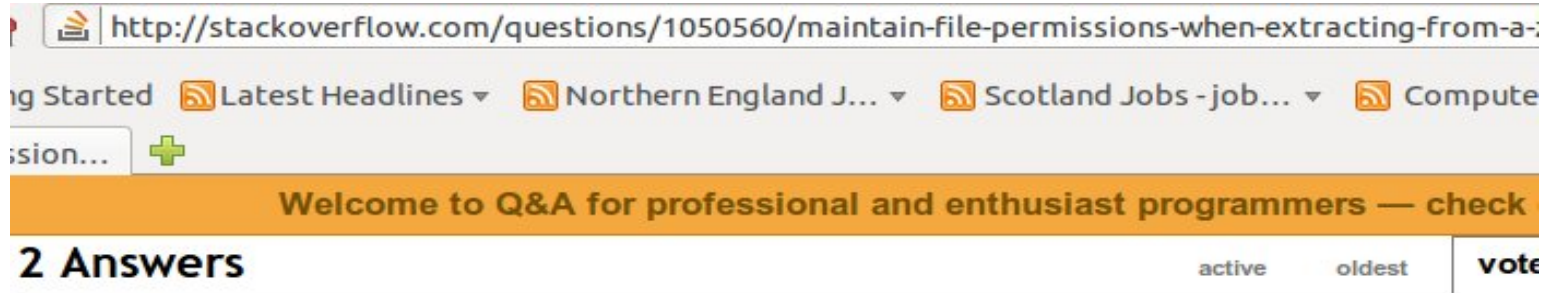
Criteria

Criteria	Related Library of Congress SF	Description
Completeness	N/A, only discussion about available tools	access to full syntactic, semantic and pragmatic information
Recoverability	N/A, only discussion about not using compression and encryption	localised damage control
Validation	Disclosure	e.g. piecewise hashing
Scalability	N/A	Influence on process efficiency
Transparency	Disclosure, Impact for patents, Transparency	e.g. open to direct analysis with basic tools
Metadata flexibility	Self documentation	user defined metadata
Flexible data handling	External dependencies	restrictions on source, size, and type of platform

Library of Congress sustainability factors from:
<http://www.digitalpreservation.gov/formats/sustain/sustain.shtml>

Comparison of formats (1)

Completeness of data



http://stackoverflow.com/questions/1050560/maintain-file-permissions-when-extracting-from-a-

Latest Headlines ▾ Northern England J... ▾ Scotland Jobs - job... ▾ Comput...

Welcome to Q&A for professional and enthusiast programmers — check

2 Answers active oldest vote



I think it is actually impossible to keep the permissions correctly.

4

Permissions are very OS specific: while POSIX file permissions allow the user to set whether you can read, write or execute a file for the file owner, the group and others, the NTFS file system has a similar system but the concept for an execute permission is in-existent. And the early FAT/FAT32 file system, do not have file permissions at all (a part from the read-only attribute).

Being cross-platform, it would be difficult for java to set the permission properly on the newly created (unzipped) files depending on the underlying OS....

That said, Java 6 has a new `java.io.File` class that allows you to set permissions (with methods like `setExecutable()`, `setReadable()`, etc... see <http://java.sun.com/javase/6/docs/api/java/io/File.html>)

These helped me a lot, especially the `setExecutable()` which was my main concern when having to unzip executables on a Linux file system. And you don't have to bother about figuring out what OS you are running on as the method will simply do nothing if running under Windows or other systems without the concept of executable files.

Hope that helps

Recovery functions

Tar:

Native recovery option exists, the utility cpio handles recovery of corrupt tar, Tar Repair tool exists for windows, and the availability of Advancerpair.

7z:

None found at the time of investigation. WinRar utilities might help.

PeaZip:

None found at the time of investigation.

Validation mechanisms

Tar:

No hashing utility.

7-zip:

CRC and SHA-256

PeaZip:

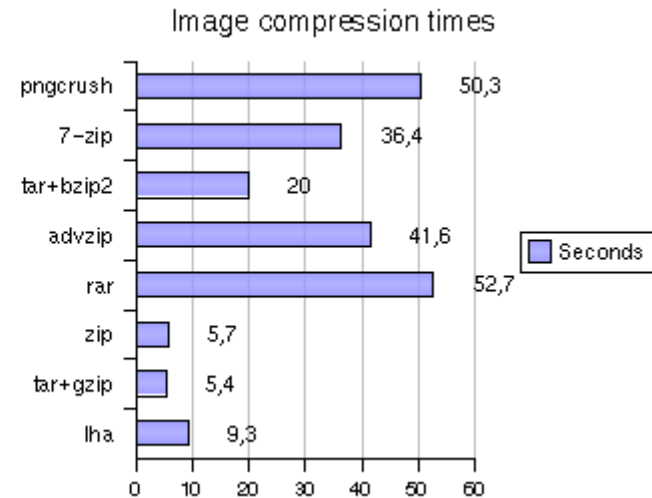
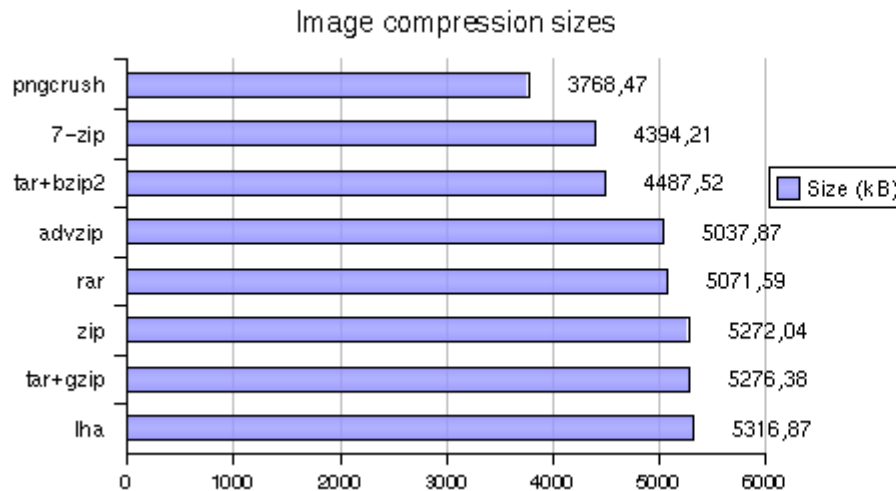
15 algorithms available (Adler32, CRC16/24/32/64, eDonkey, MD4, MD5, Ripemd160, SHA1, SHA224/256/386/512, Whirlpool512)

Only PeaZip performs piecewise hashing.

Comparison of formats (4)

Scalability:

Text compression: 7-zip, PeaZip, tar+gzip, tar+bzip2



© Copyright 2004 Juha Nieminen

<http://warp.povusers.org/ArchiverComparison/>

Tar does not allow random access of files, but requires the smallest amount of memory to process files.

Metadata flexibility

Tar, **7-zip**, and **Peazip** can only contain predefined basic metadata such as

- file name,
- file mode,
- owner's numeric user id,
- group's numeric user id,
- file size in bytes,
- last time modified in numeric Unix time format,
- checksum (if possible to calculate),
- link indicator (file type),
- name of linked file.

PeaZip may not restore all of these attributes on OS X. **7-zip** resets permissions across platforms.

Flexibility handling data

Splitting files:

Tar and 7-zip cannot control segment size nor create distributed storage.
PeaZip can split files.

Size limits:

7-zip limits file size to 16 exabytes in file size and a memory limit of 4 gigabytes on 32-bit systems and requirement for large memory in general.
PeaZip limits file size to 2^{64} bytes – while there is a limit on memory, the limit is not clear.

File system, object type and platform requirements:

PeaZip imposes limitations on file systems.

Comparison of formats (7)

Main weakness of packaging formats designed for backup or transfer of files:

1. Coverage of data is incomplete
2. Lack of recovery and validation mechanisms (tar is supported by the most number of tools)
3. Lack of metadata flexibility and unreliable recovery of metadata across platforms.
4. Variable limits on file size, memory and file systems.

Main advantages of WARC and AXF format:

1. flexible metadata handling

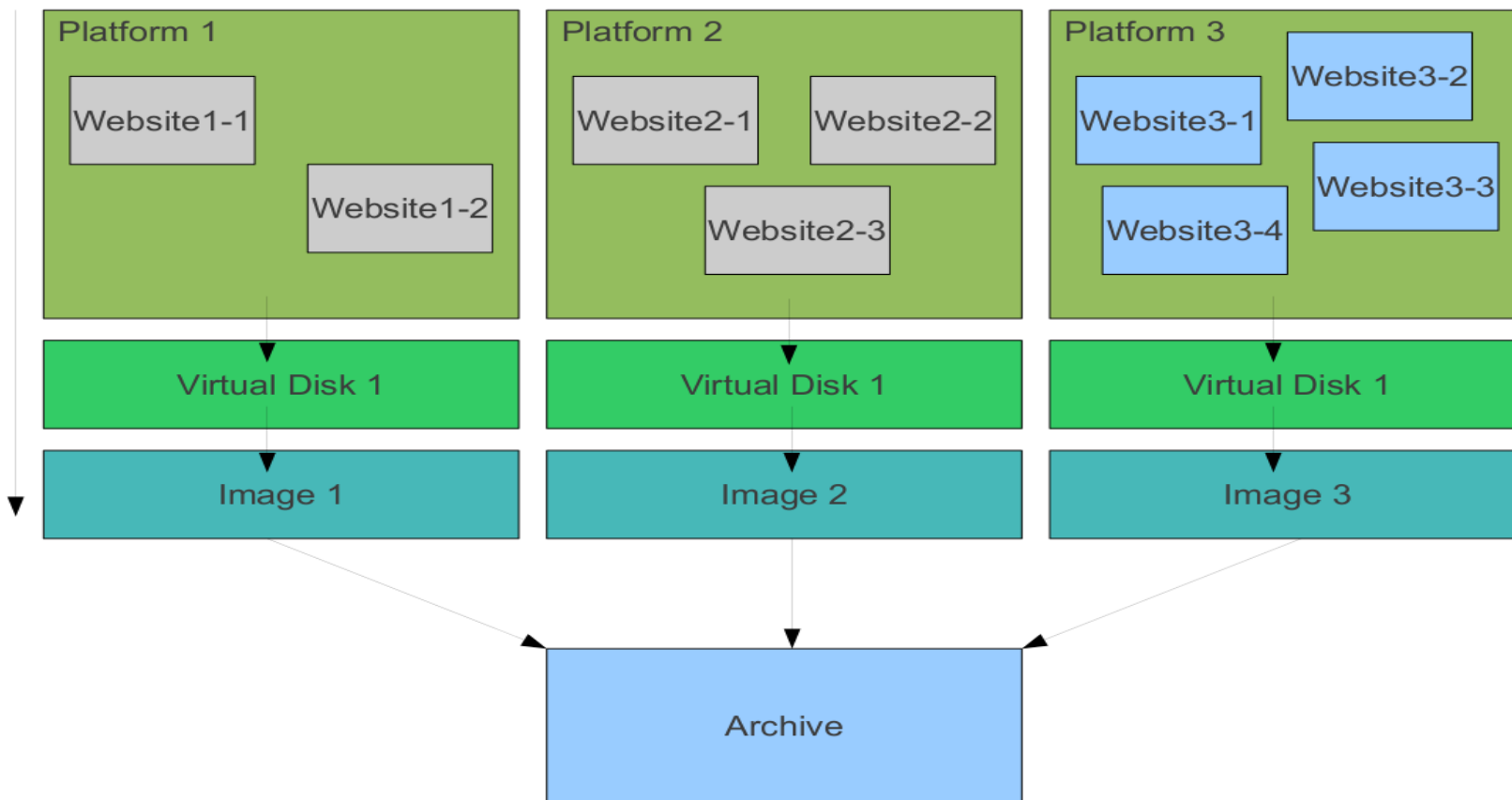
However, completeness of data is unclear, and platform independence is unclear (e.g. Wayback machine is required for WARC).

Main disadvantage of disk image formats: the input has to be a entire disk!

Tar, WARC, aff

Attribute	tar	WARC	aff
Completeness	partial File structure preserved but not other dependencies and change history.	no	yes
Recoverability	no	yes	yes
In-built validation	possible with gzip	no	yes
Scalability	no Have to unpack everything before it can be searched or indexed. May have limits on size if it becomes huge.	partial No information on whether it can be searched without unpacking and decompressing.	yes
Transparency	yes	yes	yes
Flexibility embedding metadata	no	yes	yes
Flexibility handling data	partial Cannot control file sizes. Access possible using several software, but, software might be proprietary.	partial Rendered accessed only by Internet archive software. As it does not interact with embedded data, size may be difficult to control	partial Input data only in the form of disks. Easy manipulation of data chunk size. Access possible using several access software.

Overcoming the limitations of *aff*



We have:

1. discussed attributes for file formats that need to be considered within an archive to support digital preservation;
2. compared a broad range of file formats with respect to seven core file format attributes that we have identified;
3. made a direct comparison of three of the file formats, tar, WARC, and aff; and,
- 4 proposed the Advanced Forensic File (aff) format, as the most robust among the three formats as a data-mining aware preservation storage format for a web archive.

HOME

CONTACT

ABOUT

bulk_extractor

bulk_extractor is a C++ program that scans a disk image, a file, or a directory of files and extracts useful information without parsing the file system or file system

structures. The results are stored in *feature files* that can be easily inspected, parsed, or processed with automated tools.

bulk_extractor also created a histograms of features that it finds, as features that are more common tend to be more important.

In addition to **bulk_extractor**, we have made available a small number of python programs that perform automated processing on the feature files.

bulk_extractor is distinguished from other forensic tools

by its *speed* and *thoroughness*. Because it ignores file system structure, **bulk_extractor** can process different parts of the disk in parallel. In practice, the program splits the disk up into 16MiByte pages and processes one page on each available core. This means that 24-core machines process a disk roughly 24 times faster than a 1-core machine. **bulk_extractor** is also thorough. That's because **bulk_extractor** automatically detects, decompresses, and recursively re-processes compressed data that is compressed with a variety of algorithms. Our testing has shown that there is a significant amount of compressed data in the unallocated regions of file systems that is missed by most forensic tools that are commonly in use today.

Another advantage of ignoring file systems is that **bulk_extractor** can be used to process any digital media. We have used the program to process hard drives, SSDs, optical media, camera cards, cell phones, network packet dumps, and other kinds of digital information.

© Copyright 2011 AFFLIB. All rights reserved.

Post-Processing

We have developed four programs for post-processing the **bulk_extractor** output:

bulk_diff.py

This program reports the differences between two **bulk_extractor** runs. The intent is to image a computer, run **bulk_extractor** on a disk image, let the computer run for a period of time, re-image the computer, run **bulk_extractor** on the second image, and then report the differences. This can be used to infer the user's activities within a time period.

cda_tool.py

This tool, currently under development, reads multiple **bulk_extractor** reports from multiple runs against multiple drives and performs a multi-drive correlation using Garfinkel's Cross Drive Analysis technique. This can be used to automatically identify new social networks or to identify new members of existing networks.

identify_filenames.py

In the **bulk_extractor** feature file, each feature is annotated with the byte offset from the beginning of the image in which it was found. The program takes as input a **bulk_extractor** feature file and a DFXML file containing the locations of each file on the drive (produced with Garfinkel's fiwalk program) and produces an annotated feature file that contains the offset, feature, and the file in which the feature was found.

make_context_stop_list.py

Although forensic analysts frequently make "stop lists"—for example, a list of email addresses that appear in the operating system and should therefore be ignored—such lists have a significant problem. Because it is relatively easy to get an email address into the binary of an open source application, ignoring all of these email addresses may make it possible to cloak email addresses from forensic analysis. Our solution is to create *context-sensitive stop lists*, in which the feature to be stopped is presented with the context in which it occurs. The `make_context_stop_list.py` program takes the results of multiple **bulk_extractor** runs and creates a single context-sensitive stop list that can then be used to suppress features when found in a specific context. One such stop list constructed from Windows and Linux operating systems is available on the bulk extractor website.

© Copyright 2011 AFFLIB. All rights reserved.

fiwalk

fiwalk is a program that processes a disk image using the SleuthKit library and outputs its results in Digital Forensics XML, the Attribute Relationship File Format (ARFF) format used by the Weka Datamining Toolkit, or an easy-to-read textual format.

The fiwalk source code comes with **fiwalk.py**, a Python module that makes it easy to create digital forensics programs. Also included are several demonstration programs that use **fiwalk.py**:

- **iblkfind.py** – Given a disk block in a disk image, this program tells you which file(s) map that sector.
- **icarvingtruth.py** - Given two or more images of the same disk at different points in time, this program finds files that are present in the earlier images that can only be recovered from the later images using file carving techniques.
- **idifference.py** - Given two or more images of the same disk at different points in time, this program tells you what changes took place between each one.
- **extract.py** - Allows the extraction of files that match a particular pattern.
- **igrep.py** - Searches every file in a disk image for a particular string. When found, prints, the file and the offset within the file that the string was found.
- **ihistogram.py** – Prints a histogram of file types found in the disk image.
- **imap.py** - Displays a “map” of where files are present in the disk image.
- **imicrosoft_redact.py** - Modifies a disk image of a bootable Microsoft operating system so that the image can no longer be boot and so that any Microsoft copyrighted file in the \Windows directory cannot be executed. This allows the disk image of a Microsoft operating system to be distributed without implicitly violating Microsoft's copyright.
- **iredact.py** - An experimental disk redaction program which allows the removal of specific files matching specific criteria.
- **iverify.py** – Given a disk image and a previously created XML file, verifies that each file in the DFXML file is still present in the disk image.
- **sanitize_xml.py** - Given a DFXML file, sanitize file names so that no personally identifiable information is leaked if the DFXML file is distributed.

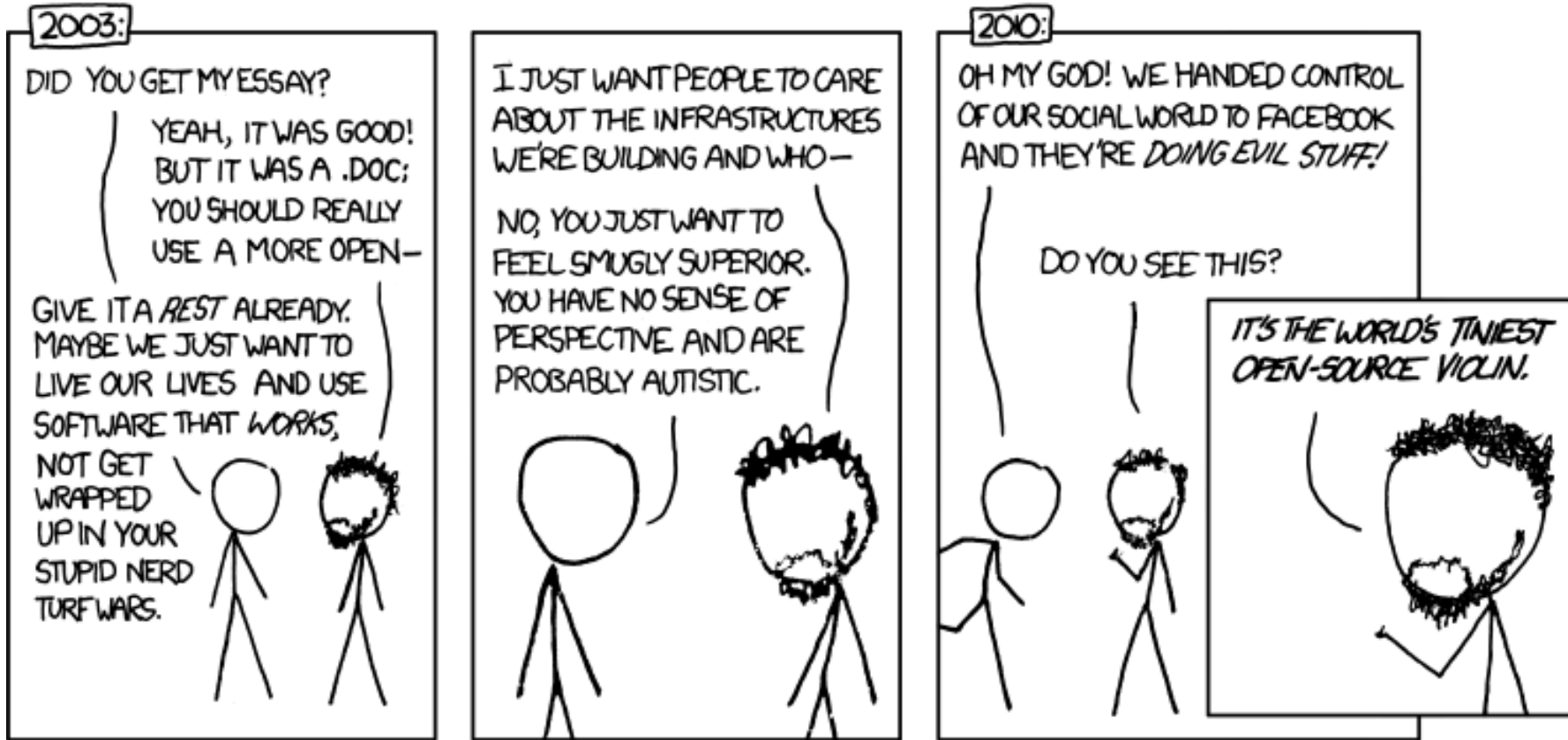
Further studies required:

1. how will information be captured into virtual disks (e.g. will blogs from one website be kept together?), and
2. how will the information within each object be segmented and distributed?

Next step:

A small-scale experiment to compare the target formats with respect to selected blogs harvested from the web, using the identified preservation attributes as evaluation criteria.

Thank you



Permanent link to this comic: <http://xkcd.com/743/>

Image URL (for hotlinking/embedding): <http://imgs.xkcd.com/comics/infrastructures.png>

This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. This means you're free to copy and share these comics (but not to sell them).