Forensic Analysis of Internet Explorer Activity Files

by Keith J. Jones
keith.jones@foundstone.com

3/19/03

Table of Contents

1. Introduction _____ *4*

2. The Index.dat File Header _____ *6*

3. The HASH Table _____ *10*

4. The Activity Records _____ *14*

4.1. The URL Activity Record _____ **15**

4.2. The REDR Activity Record _____ **23**

4.3. The LEAK Activity Record _____ **26**

5. Deleted Activity Records _____ *27*

6. Pasco – The IE Internet Activity Parser _____ *28*

Table of Figures

Figure 1 – One location for Index.dat 4
Figure 2 – The Index.dat File Size 6
Figure 3 – The HASH Table Offset 7
Figure 4 – The Beginning of the HASH Table 7
Figure 5 – The Index.dat Directories 8
Figure 6 – The HASH Table Linked List 10
Figure 7 – The Second Hash Table 11
Figure 8 – A Valid Activity Record in the HASH Table 12
Figure 9 – A Valid Activity Record 12
Figure 10 – A URL Activity Record 15
Figure 11 – The URL Activity Record Web Site Offset 16
Figure 12 – The URL Activity Record Filename Data 16
Figure 13 – The URL Activity Record Filename Data Offset 17
Figure 14 – The URL Activity Record HTTP Header Data 17
Figure 15 – The URL HTTP Header Data Offset 18
Figure 16 – The URL Activity Record Last Modified Time Stamp 19
Figure 17 – The URL Activity Record Last Accessed Time Stamp 20
Figure 18 - Location of the Directory Number 21
Figure 19 – A REDR Activity Record 23
Figure 20 – The REDR Activity Record Length 24
Figure 21 – The URL in a REDR Activity Record 24
Figure 22 – A LEAK Activity Record 26
Figure 23 - Pasco's Output 29

Listing of Tables

Table 1 - Common Index.dat File Locations for Internet Explorer 5
Table 2 - Relevant Fields in the Index.dat File Header 9
Table 3 - Relevant Fields in the HASH Table Header 13
Table 4 - Relevant Fields in the URL Activity Record 22
Table 5 - Relevant Fields in the REDR Activity Record 25

1. Introduction

Internet Explorer is an application used to browse the web that an overwhelming majority of computer users utilize on a daily basis. One of the many challenges for the forensic analyst is to reconstruct the web browsing habits for the subject under investigation. In order to reconstruct this activity, one must analyze the internal data structures of the web browser cache files for Internet Explorer. Unfortunately, the internal structures for the cache files are not well. Additionally, publicly available tools used to reconstruct internet activity are commercial which typically makes the methods they use proprietary. This research was performed to give the computer forensic community an open source, reproducible, forensically sound, and documented method to reconstruct Internet Explorer activity. The information in this paper was determined from a simple hex editor on a sample cache file. The relevant data introduced in this paper was discovered while analyzing the internal structures for a cache file and comparing the results to known output generated from IE History (www.phillipsponder.com), a popular commercial tool to reconstruct Internet Explorer activity, on the same file.

To understand what files are relevant to us, we must give some background on Internet Explorer. Internet Explorer saves numerous files named "`index.dat`" within each user's home directory on the computer system. This file maps web sites visited to locally saved cache files in randomly named directories so that the next time the user visits the same web site, he will not have to download the same graphics and web pages all over again. The following figure illustrates where an "`index.dat`" file resides.

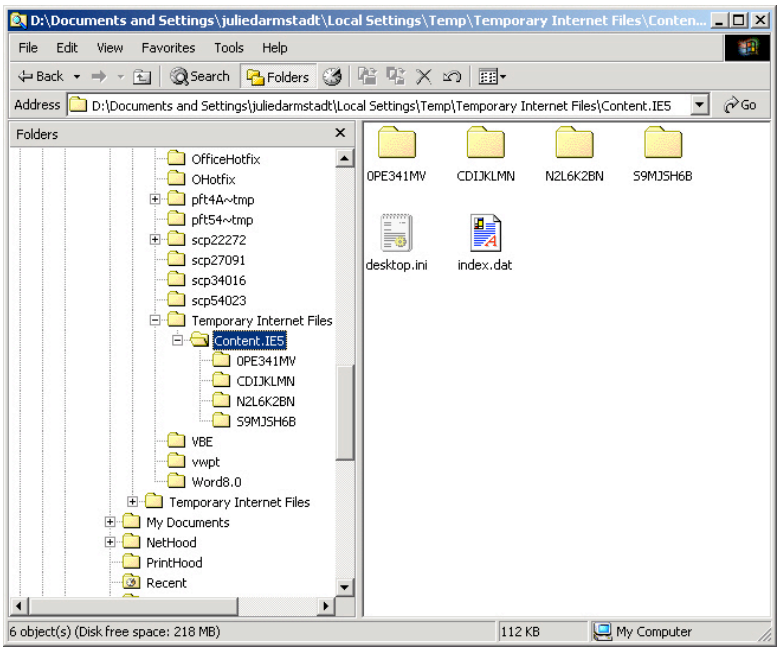


Figure 1 – One location for Index.dat

The following table lists additional areas of the file system other `index.dat` files may be located:

Table 1 - Common Index.dat File Locations for Internet Explorer

Operating System	*File Path(s)*
Windows 95/98/Me	\Windows\Temporary Internet Files\Content.IE5\ \Windows\Cookies\ \Windows\History\History.IE5\
Windows NT	\Winnt\Profiles\<username>\Local Settings\Temporary Internet Files\Content.IE5\ \Winnt\Profiles\<username>\Cookies\ \Winnt\Profiles\<username>\Local Settings\History\History.IE5\
Windows 2K/XP	\Documents and Settings\<username>\Local Settings\Temporary Internet Files\Content.IE5\ \Documents and Settings\<username>\Cookies\ \Document and Settings\<username>\Local Settings\History\History.IE5\

A forensic analyst can use the information found in the `index.dat` file to reconstruct a user's web activity. The structures identified during this analysis that were deemed relevant to constructing internet activity data will be discussed in detail further in this paper.

2. The Index.dat File Header

The `index.dat` file contains a header that harbors some information about the file and pointers to additional information within it. This section will analyze those fields.

The first field we notice is the *file size*. The file size is given in the file header immediately following the NULL (0x00) terminated version string. In this case it is "00 C0 01 00". With most of the numerical values found in the `index.dat` file, one must swap the bytes from left to right when reading the value. In the example below, the file size is 0x0001C000. This translates to a value of 114688 bytes, which is correct for the file used in this demonstration.

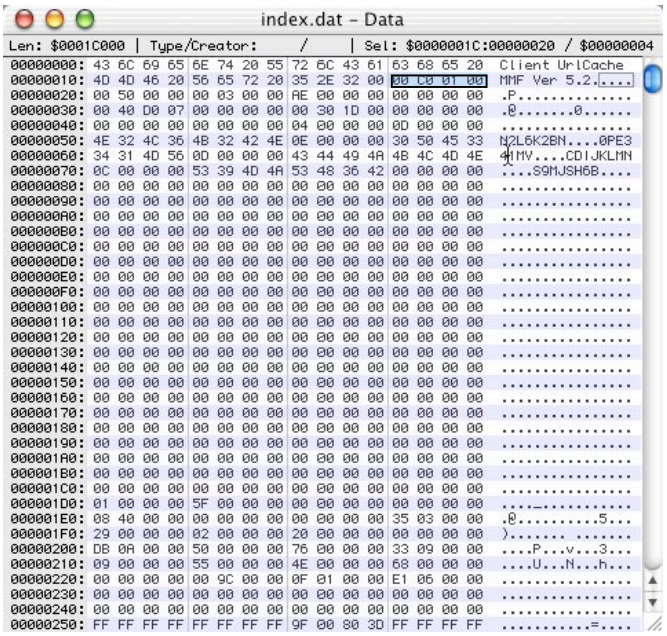


Figure 2 – The Index.dat File Size

Immediately following the file size is the location of the *HASH Table*. The HASH table is an array of data that contains entries pointing to the relevant activity data within the `index.dat` file. We will use these pointers, or offsets, to find the relevant data within the `index.dat` file. The HASH table is important enough that it will be detailed in its own upcoming section.

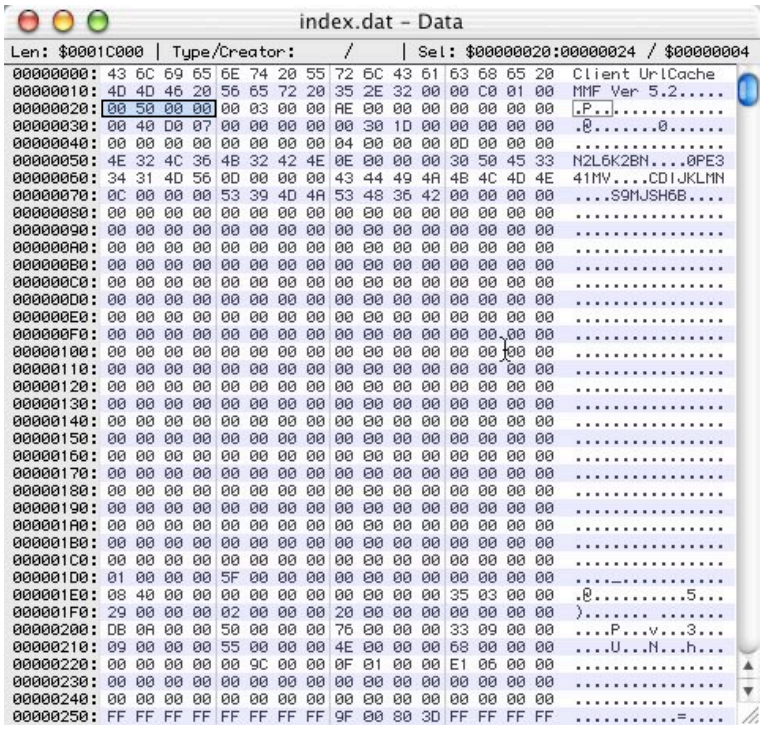


Figure 3 – The HASH Table Offset

In this case the starting value for the HASH table is "00 50 00 00" and after the byte flip translates to 0x5000. The following screen capture shows the beginning of the HASH table:

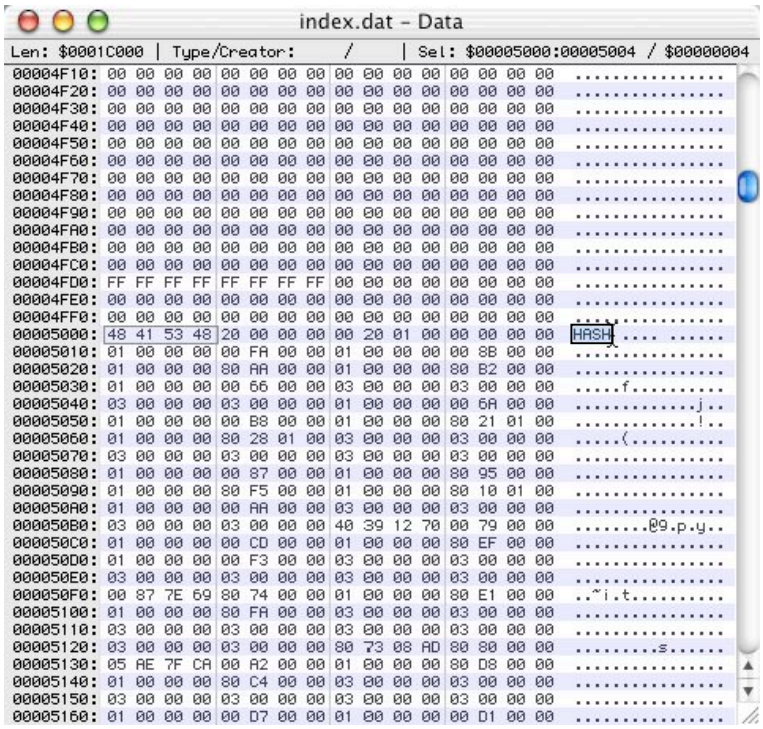


Figure 4 – The Beginning of the HASH Table

After the HASH table offset is a listing of directories that this `index.dat` file uses to store the locally cached files on the user's computer. Notice that in Figure 5 the four directories correlate exactly with Figure 1.

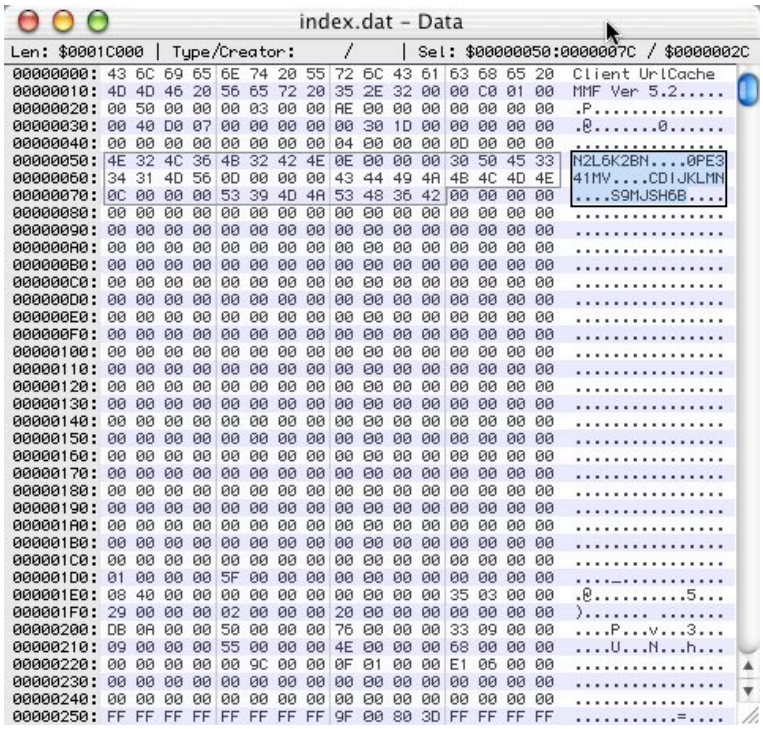


Figure 5 – The Index.dat Directories

In this case, the `index.dat` file is responsible for the following directories:

- N2L6K2BN
- 0PE341MV
- CD1JKLMN
- S9MJSH6B

These directories contain the files that were actually downloaded from the web. We can use the contents to reconstruct web pages a subject visited. This is information that is typically missing from most commercial tools used to reconstruct Internet Explorer activity.

The fields in the `index.dat` header are summarized in the following table:

Table 2 - Relevant Fields in the Index.dat File Header

Field Name	*Offset (in bytes)*	*Size (bytes)*	*Description*
File Length	0x1C	4	This field contains the length of the `index.dat` file, in 0x80 byte sized records.
HASH Table Offset	0x20	4	This field contains the offset (in bytes) for the beginning of the HASH Table.
Cache Directories	0x50	12	This field contains the directories where files are stored that make up the content of the cache. Each directory is 12 bytes long, where only the first 8 bytes are relevant.

3. The HASH Table

The HASH table is our "master lookup table" to find valid activity records within the `index.dat` file. It is very much similar to a FAT table for a file system. Furthermore, if an `index.dat` file is large enough, it can have more than one HASH table. Each HASH table contains a pointer to the next HASH table, making it a linked list. This section will discuss the important data fields within one of the HASH tables.

This first field is the length of the HASH table. Figure 4 presents the first set of 4 bytes after the name "HASH" having the value of "20 00 00 00" which translates to 0x20, or 32. Upon observation, each record with the `index.dat` file is a multiple of 0x80 (128) bytes long. Therefore, we find that the HASH table is 32*128=4096 (0x1000) bytes long. For the example given in Figure 4, the HASH table ends at 0x6000, which is the expected value.

It is important to note that there can be more than one HASH table within an `index.dat` file. The next field within the HASH table is a pointer, or offset in bytes, to the next HASH table. The next HASH table pointer will be zero for the last HASH table in the file.

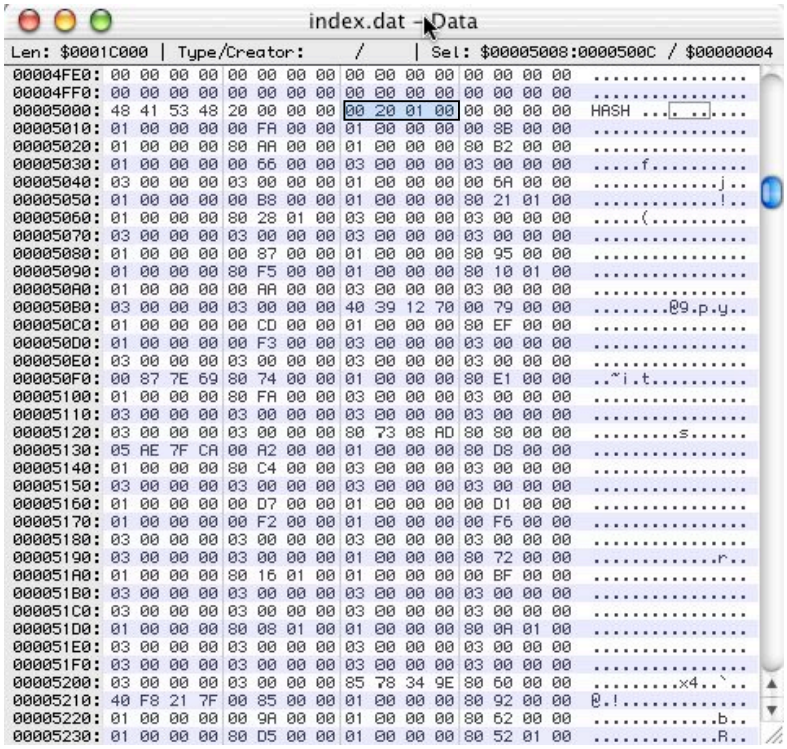


Figure 6 – The HASH Table Linked List

In this example, the next HASH table should be at "00 20 01 00", which is 0x12000 after the byte flip. Looking at offset 0x12000 from the beginning of the file shows us the next HASH table. This HASH table is empty in this example, but could contain the same structured data described in this section and linked to another HASH table, and so on.

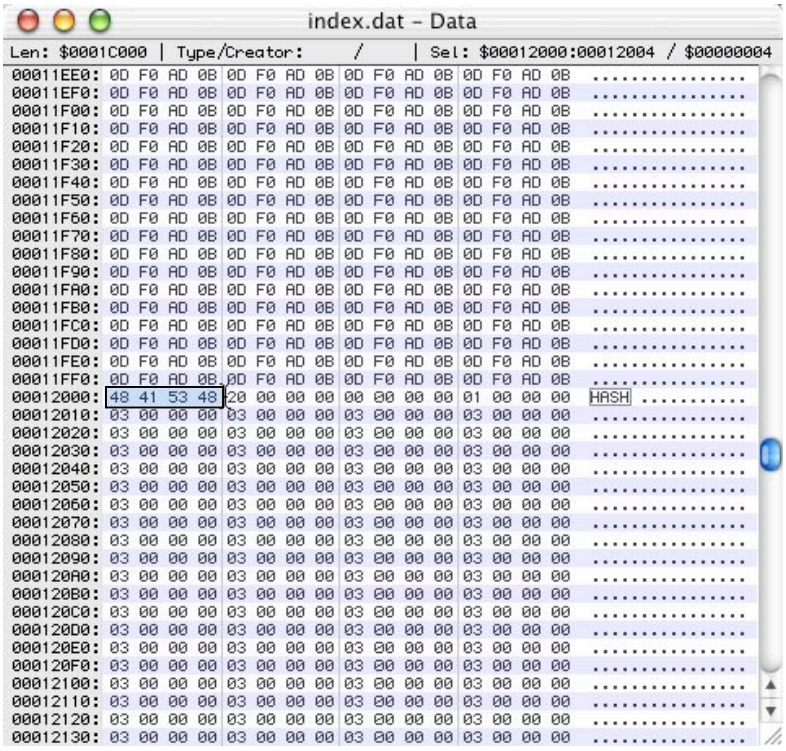


Figure 7 – The Second Hash Table

The following data in the HASH table are pointers to the relevant activity data within this history file. Each entry in the HASH table is 8-bytes long. There seems to be three unique options for the first 4 bytes of these 8:

1) The value is "01 XX XX XX", where XX can be any value.
2) The value is "03 XX XX XX", where XX can be any value.
3) The value is "YY XX XX XX", where YY is not 0x01 or 0x03 and XX can be any value.

It seems as though the second 4 bytes should point to a record containing Internet activity history. In option 1 above, the 4-byte value that immediately follows the first four bytes *does not* point to an activity record. Additionally, if the pointer is to a memory location 0x0BADF00D, then we know it is not valid. 0x0BADF00D is an invalid memory location because that value is used by default when the `index.dat` file is created and populated.

In the case of the second and third options previously presented, the second set of 4 bytes point to the start of a *valid* activity record. In Figure 8 it is shown that a valid activity record should be found at "00 A2 00 00", which is 0xA200.

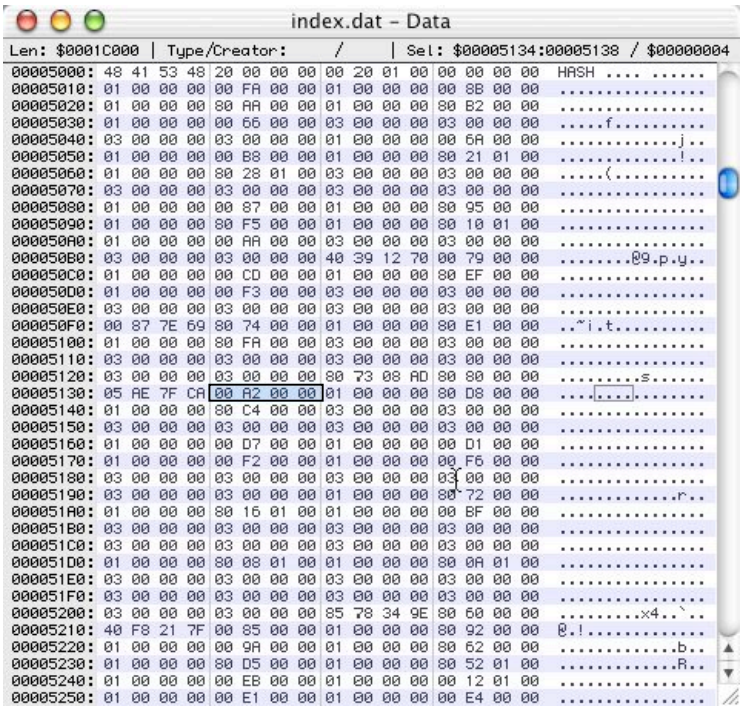


Figure 8 – A Valid Activity Record in the HASH Table

After we jump to offset 0xA200 within the file, we see that a valid[1] activity record is present.

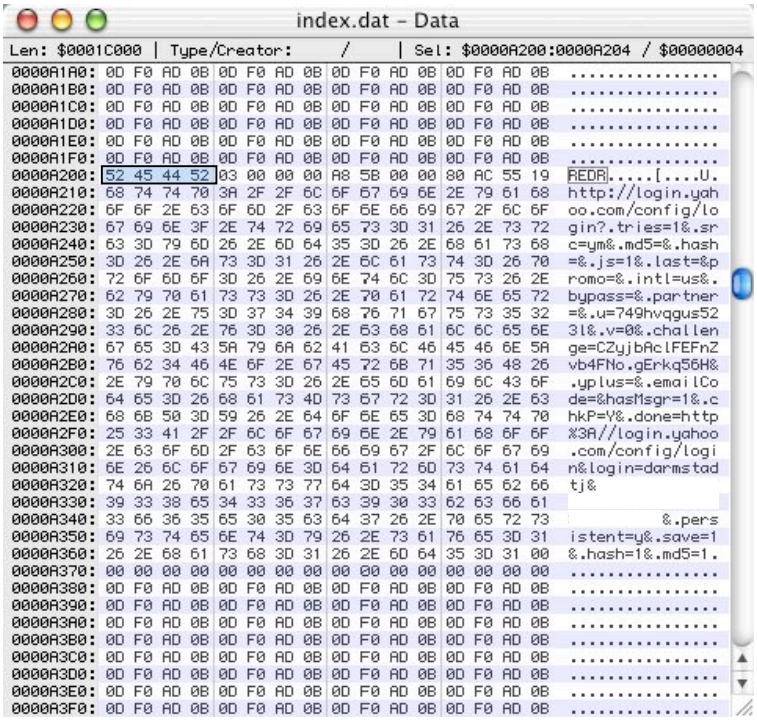


Figure 9 – A Valid Activity Record

[1] A valid activity record will be clearly defined later in this paper.

The relevant fields in the HASH table are summarized in the following table:

Table 3 - Relevant Fields in the HASH Table Header

Field Name	*Offset (in bytes) from the beginning of the HASH Table*	*Size (bytes)*	*Description*
HASH Table Length	4	4	This field contains the length of the HASH Table, in 0x80 byte blocks.
Next HASH Table	8	4	This field contains the offset (in bytes, from the beginning of the file) where the next HASH Table can be located. If the value is zero, this is the last HASH Table in the `index.dat` file.
Activity Record Flags	16 + 8*n; where n=0,1,2,3…	4	This 4 byte field contains the flags for the activity record. If the first byte equals 01, the following field does not represent a valid activity record pointer.
Activity Record Pointers	20 + 8*n; where n=0,1,2,3…	4	This is an activity record offset, in bytes, from the beginning of the file.

4. The Activity Records

The activity records contain the main information we are attempting to recover from the `index.dat` file. The activity records follow a generic structure type:

TYPE, LENGTH, DATA

- The "TYPE" field contains some of the following activity types and is 4 bytes in length:
    - REDR
    - URL
    - LEAK
- The "LENGTH" field contains the length, measured in 0x80 (128) byte sized blocks, of the activity record. The "LENGTH" field is 4 bytes long.
- The "DATA" field is dependent upon the *type* of activity record we are analyzing. The most common types and what values exist in the DATA field will be discussed in the following subsections.

4.1. *The URL Activity Record*

The URL activity record is a set of data that represents a URL, or website, a user visited. Figure 10 is an example of one such record.

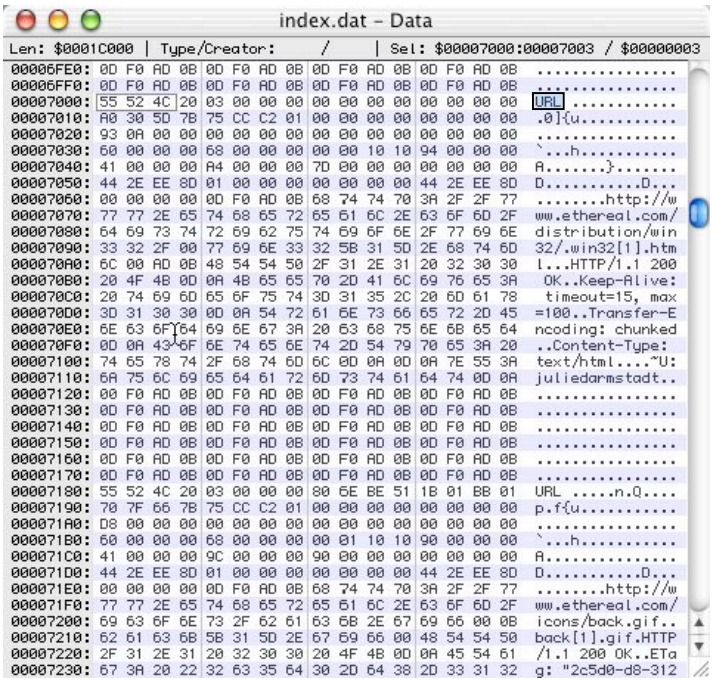


Figure 10 – A URL Activity Record

We see that this record reports a length of "03 00 00 00", or 0x03 blocks of 0x80 bytes in size. This is 0x03*0x80=0x180 bytes which makes sense because we see the next URL activity record starts at offset 0x7180. Next, we see that the actual URL the user visited is located at offset 0x68 from the beginning of the activity record. Observe that the offset of this URL is located at 0x34 bytes (see Figure 11) from the beginning of the activity record. Therefore, we must first read the value at offset 0x34 and jump to that position in the activity record to read the NULL terminated URL string.

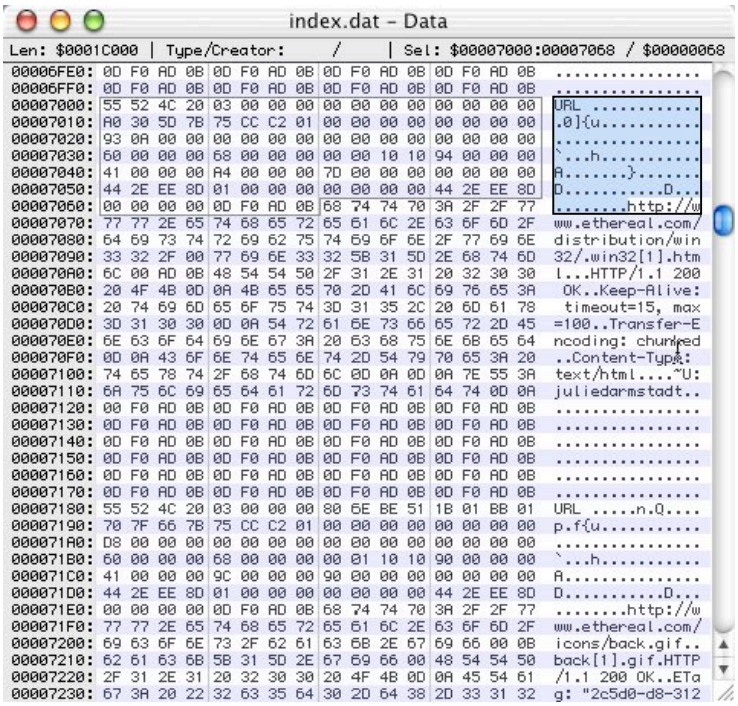


Figure 11 – The URL Activity Record Web Site Offset

We know that the URL can be variable length and strings such as this are terminated by a NULL (0x00) byte. Therefore, to quickly look up the fields that come after the URL there must be an offset somewhere in the activity record's header. If we look at the file name for the locally cached file stored on the hard disk, we see that it is 0x94 bytes from the beginning of the activity record.

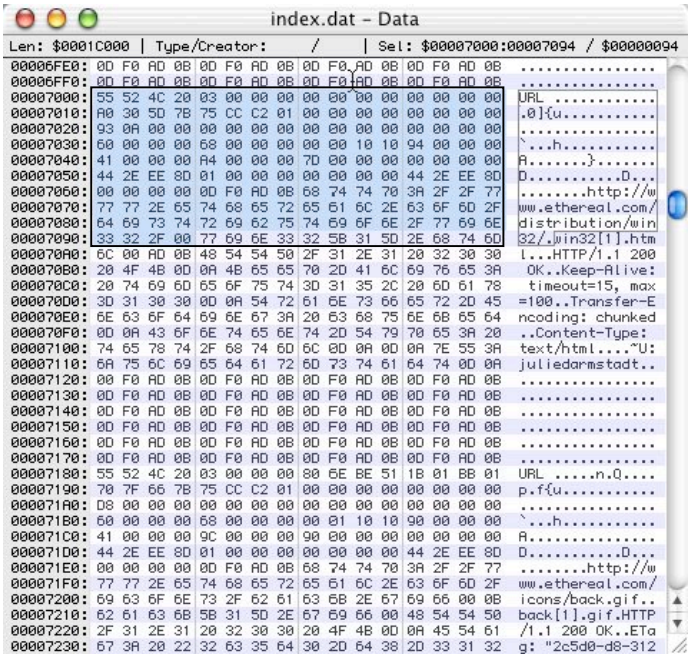


Figure 12 – The URL Activity Record Filename Data

In searching through the header of the activity record, we see “94 00 00 00” (0x94) exists 0x3C bytes from the beginning of the activity record.

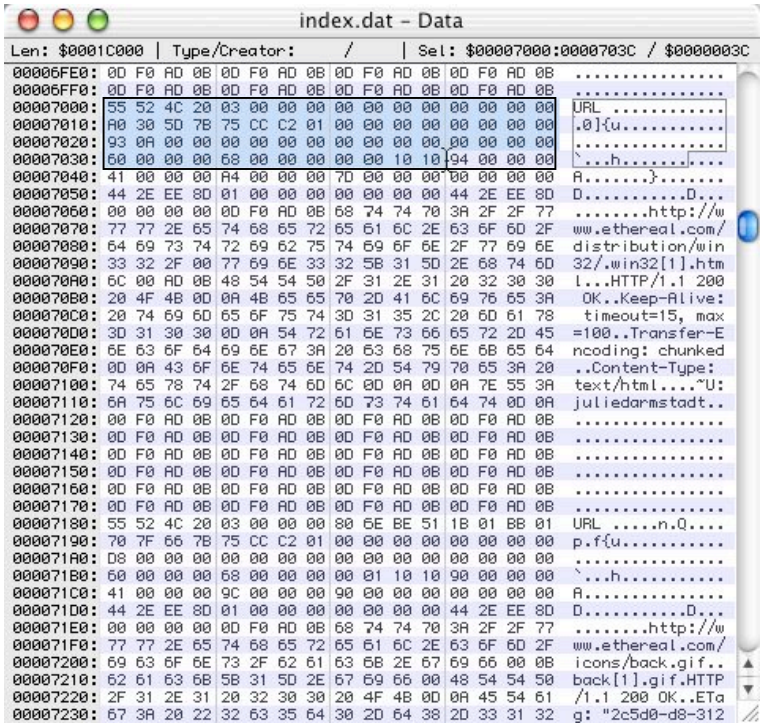


Figure 13 – The URL Activity Record Filename Data Offset

Similarly, the HTTP headers are at offset 0xA4:

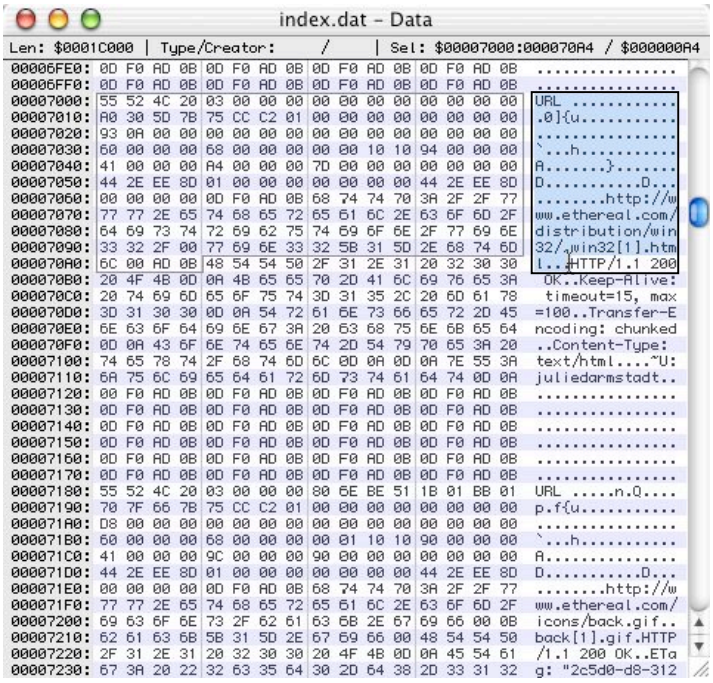


Figure 14 – The URL Activity Record HTTP Header Data

The HTTP header offset “A4 00 00 00” (0xA4) is 0x44 bytes from the beginning of the activity record.

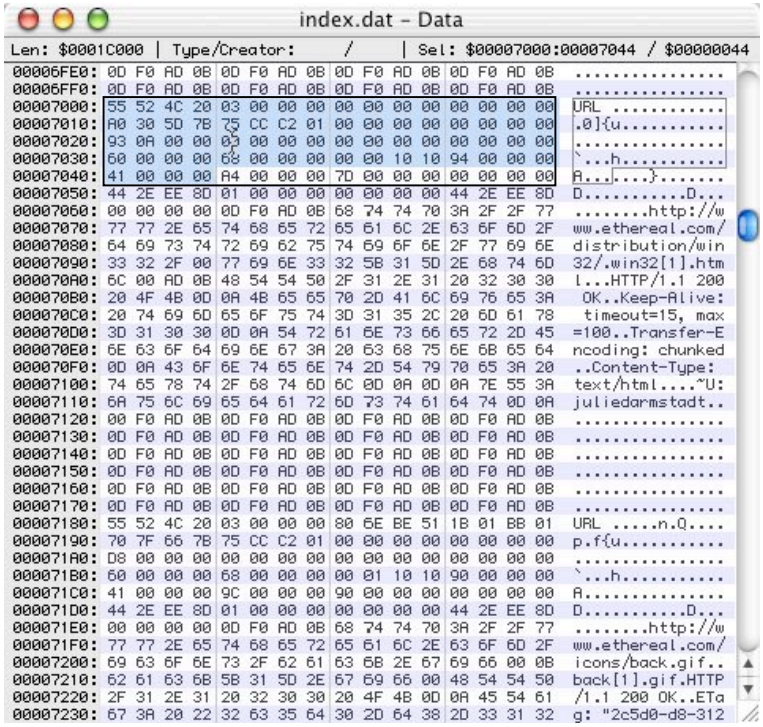


Figure 15 – The URL HTTP Header Data Offset

Two other important fields we would want to know when reconstructing a subject’s Internet activity are last modified and last accessed time stamps. The last modified time stamp would be when the information was changed on the web server. The last accessed time stamp would be when the last time Internet Explorer visited the URL. Both of these fields are found directly after the length of the activity record and are 8-byte values each. The last modified field is found first:

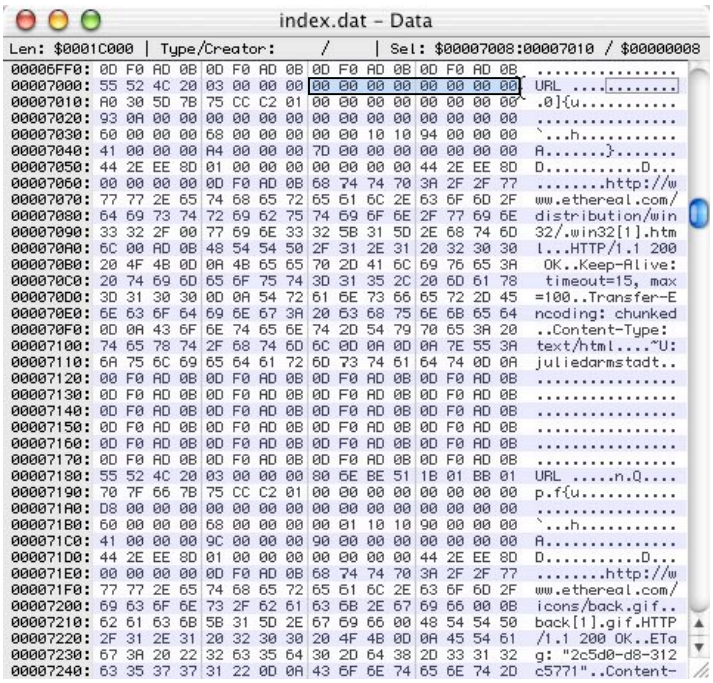


Figure 16 – The URL Activity Record Last Modified Time Stamp

Unfortunately, the example we've been using so far was not a good one. We see this activity record has "00 00 00 00 00 00 00 00" as the last modified time. However, if we look at the next activity record in Figure 16, we see its last modified time was "80 6E BE 51 1B 01 BB 01" or 0x01BB011B51BE6E80. The fact that one activity record has all zeros for a last modified time stamp is not important to us, as an investigator, because we do not care when the web server last updated its content. For most Internet activity reconstruction attempts, we are interested in the last time someone accessed a web page. The last accessed field is found in the next 8-byte field:

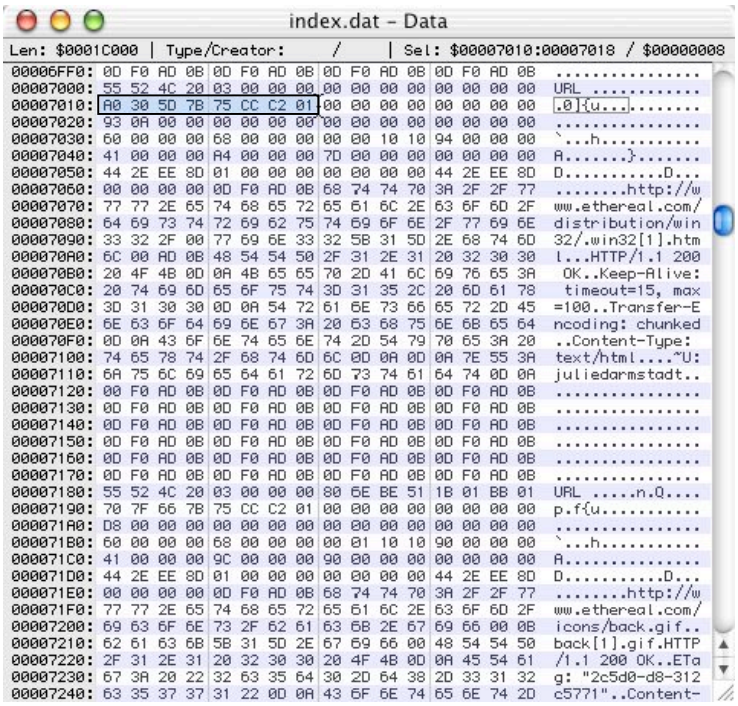


Figure 17 – The URL Activity Record Last Accessed Time Stamp

Now that we know which fields are time stamps we must translate them to something a human can understand. Windows saves time stamps in what has been defined as "FILETIME" format. FILETIME format is the number of ticks, in 100ns increments, since 00:00 1 Jan, 1601 (UTC). Since the rest of the world uses the Unix definition of time, which is the number of seconds since 00:00 1 Jan 1970, we must be able to translate the FILETIME format to the Unix time format. This is done with the following simple equation:

$$(Unix\ Time) = A * (NT\ Time) + B$$

Since the ticks in FILETIME are at 100ns intervals, we know that "A" is 10^{-7}. The trick is finding "B". "B" is the number of seconds between 1 Jan 1601 and 1 Jan 1970. We do not have to painstakingly calculate that value because it is well documented with MSDN and open source initiatives that "B" is 11644473600.

The last piece of information that may be useful is in which directory, from Figure 5, the locally cached filename discovered in Figure 12 resides. Experimentation shows that that value is found at 0x39 bytes from the beginning of the activity record.

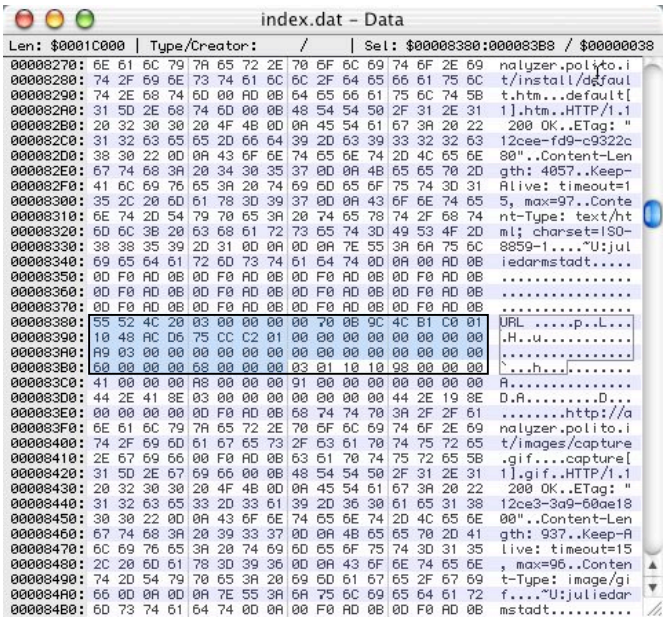


Figure 18 - Location of the Directory Number

In the example above, the “capture[1].gif” file was located within the “S9MJSH6B” directory. This is consistent because we see 0x03 at offset 0x38 from the beginning of the activity record. The value 0x03 says the file is located in the fourth folder because the first folder starts with an index of zero. The fourth folder in Figure 5 is “S9MJSH6B” and the results are consistent.

The following table summarizes the relevant fields within the URL activity record:

Table 4 - Relevant Fields in the URL Activity Record

Field Name	*Offset (in bytes) from the beginning of the URL Activity Record*	*Size (bytes)*	*Description*
Record Type	0x0	4	This is the field that contains the string "URL".
Record Length	0x4	4	This is the number of 0x80 byte blocks that the URL record contains.
Last Modified Time Stamp	0x08	8	This is the Last Modified time stamp, in FILETIME format.
Last Accessed Time Stamp	0x10	8	This is the Last Accessed time stamp, in FILETIME format.
URL Offset	0x34	4	This is the URL Offset, from the beginning of the record.
Filename Offset	0x3C	4	This is the Filename Offset, from the beginning of the record.
Local Cache Directory Index	0x38	1	This is the index (starting with zero) of the local directories containing the cache files.
HTTP Header Offset	0x44	4	This is the offset, from the beginning of the record, where the HTTP Headers are located.

4.2. *The REDR Activity Record*

The REDR type of activity record is very simple because it is just a statement of when the subject's browser was redirected to another site. The generic *TYPE, LENGTH, DATA* format still holds true for the REDR activity record. The following figure shows a REDR activity record:

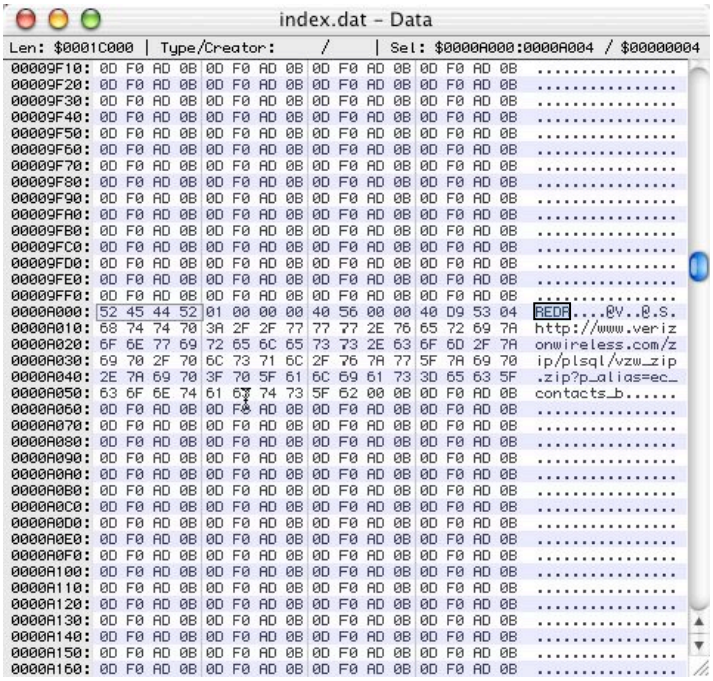


Figure 19 – A REDR Activity Record

The length of this example is "01 00 00 00" which is 0x01. That makes this record 0x80, or 128, bytes long.

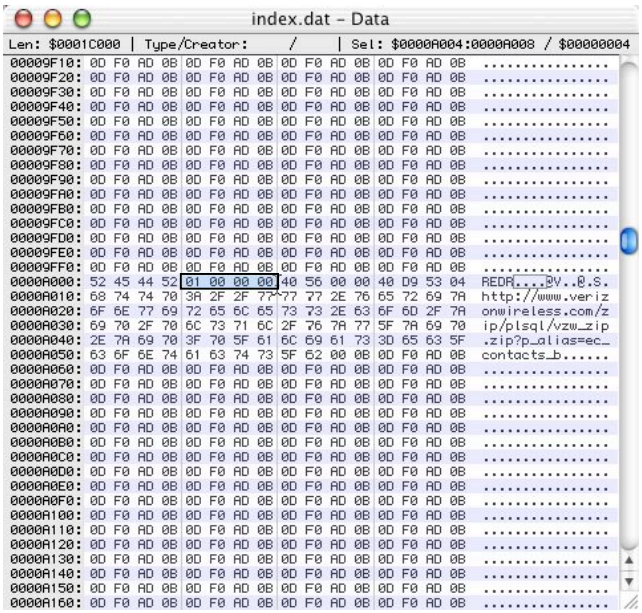


Figure 20 – The REDR Activity Record Length

The next 8-byte field would seem to be a time stamp if it were similar to the URL activity records. We know that is not the case because the right most byte (the most significant byte after the flip) is "04" and it should be a "01" to fit in with this example (This was figured out by knowing that all of the web sites listed in the sample `index.dat` file were visited within the same day). Therefore, this field is probably flag values or similar data. Lastly, the URL is located at offset 0x10 from the beginning of the record and is NULL terminated with a 0x00 byte.

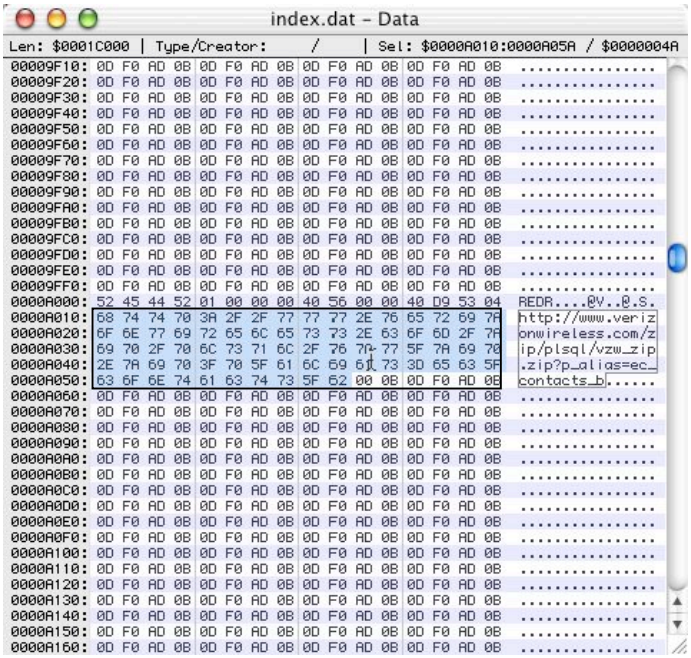


Figure 21 – The URL in a REDR Activity Record

The following table summarizes the relevant fields in the REDR activity Record:

Table 5 - Relevant Fields in the REDR Activity Record

Field Name	*Offset (in bytes) from the beginning of the URL Activity Record*	*Size (bytes)*	*Description*
Record Type	0x00	4	This is the field that contains the "REDR" string.
Record Length	0x04	4	This is the field that contains the number of 0x80 byte sized blocks that make up the REDR record.
URL	0x10	*Variable*	This is the URL, terminated by a NULL (0x00) character.

4.3. *The LEAK Activity Record*

The LEAK activity record has exactly the same internal structure as the URL activity record. At the time this document was written, it is still difficult to tell the difference between a "URL" and a "LEAK" activity record other than the different value for the *TYPE* in the header.

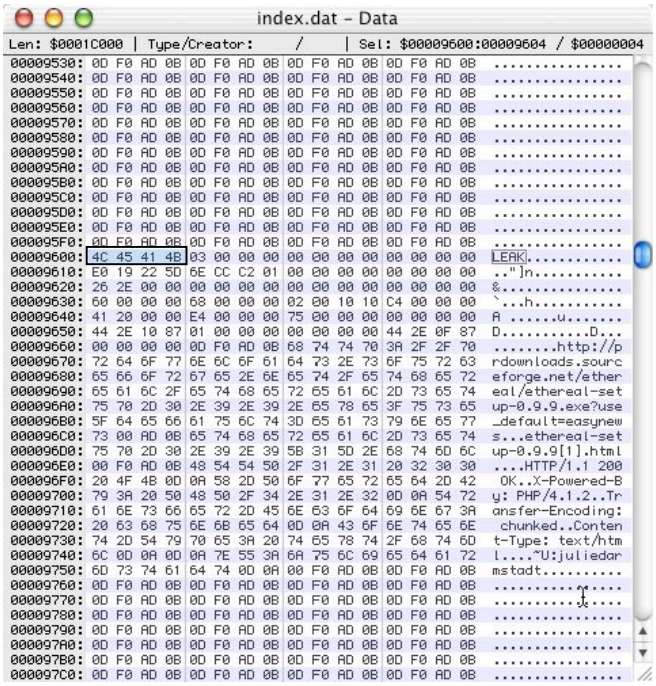


Figure 22 – A LEAK Activity Record

5. Deleted Activity Records

We know that each record is a multiple of 0x80 bytes. Knowing this, if the first four bytes (the type) were compared against the known types of activity records listed previously in this paper (URL, REDR, LEAK), it would be logical that we would be able to reconstruct any deleted or unlinked records. Through experimentation we were able to determine activity records did in fact exist even though they did not contain entries in the HASH tables. Additionally, the output of IE History did not contain the number of activity records that we recovered using the undeletion method described in this section.

6. Pasco – The IE Internet Activity Parser

Now that we have a methodology to reconstruct Internet Explorer activity from the internal data structures within an `index.dat` file, we can develop a tool to automate everything we have done by hand so far. The author developed a tool called *Pasco*, the Latin word for "browse", to do just that. Pasco is run against an `index.dat` file retrieved from a user's computer and the output is delimited text so that the investigator may import the results into his spreadsheet of choice.

Pasco can be run in two different modes: the *standard* methodology outlined in this paper (which is the default processing for Pasco), or in an *undeletion* mode. The undeletion mode ignores the information in the HASH table and reconstructs any valid activity records at every 0x80 byte boundry. *This mode may retrieve activity that was previously unreported by other tools and methods*.

The command line usage for Pasco is relatively simple:

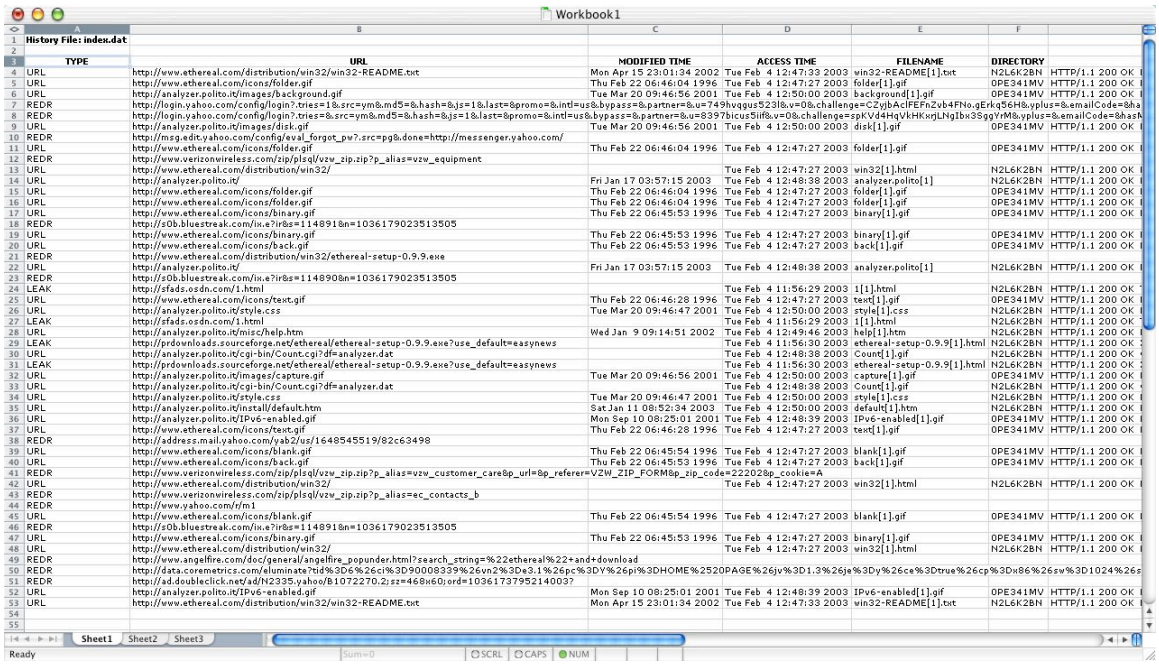
```
[kjones: pasco] kjones% ./pasco

Usage:  pasco [options] <filename>
        -d Undelete Activity Records
        -t Field Delimiter (TAB by default)
```

The "-d" option enables the undeletion mode. The "-t" option will allow the investigator to change the field delimiter. The output will be sent to standard out (the console) by default. It is suggested that Pasco is run in the following manner:

```
./pasco index.dat > index.txt
```

Once index.txt is created, the results can be imported into a spreadsheet like Microsoft Excel for further viewing, sorting, and formatting:



History File: index.dat					
TYPE	URL	MODIFIED TIME	ACCESS TIME	FILENAME	DIRECTORY
URL	http://www.ethereal.com/distribution/win32/win32-README.txt	Mon Apr 15 23:01:34 2002	Tue Feb 4 12:47:33 2003	win32-README[1].txt	N2L6K2BN
URL	http://www.ethereal.com/icons/folder.gif	Thu Feb 22 06:46:04 1996	Tue Feb 4 12:47:27 2003	folder[1].gif	0PE341MV
URL	http://analyzer.polito.it/images/background.gif	Tue Mar 20 09:46:56 2001	Tue Feb 4 12:50:00 2003	background[1].gif	0PE341MV
REDR	http://login.yahoo.com/config/login?.tries=1&.src=ym&.md5=&.hash=&.js=1&.last=&promo=&.intl=us&.bypass=&.partner=&.u=749hvqgus523l&.v=0&.challenge=CZyjbAclFEFnZvb4FNo.gErkq56Hb.vplus=&.emailCode=&ha				
REDR	http://login.yahoo.com/config/login?.tries=8&.src=ym&.md5=&.hash=&.js=1&.last=&promo=&.intl=us&.bypass=&.partner=&.u=8397bicus5if8&.v=0&.challenge=spKVd4HqVkHKxrjLNgIbx3SggYrM&.vplus=&.emailCode=&hasN				
URL	http://analyzer.polito.it/images/disk.gif	Tue Mar 20 09:46:56 2001	Tue Feb 4 12:50:00 2003	disk[1].gif	0PE341MV
REDR	http://msg.edit.yahoo.com/config/eval_forgot_pw?.src=pg&.done=http://messenger.yahoo.com/				
URL	http://www.ethereal.com/icons/folder.gif	Thu Feb 22 06:46:04 1996	Tue Feb 4 12:47:27 2003	folder[1].gif	0PE341MV
REDR	http://www.verizonwireless.com/zip/plsql/vzw_zip.zip?p_alias=vzw_equipment				
URL	http://www.ethereal.com/distribution/win32/		Tue Feb 4 12:47:27 2003	win32[1].html	N2L6K2BN
URL	http://analyzer.polito.it/	Fri Jan 17 03:57:15 2003	Tue Feb 4 12:48:38 2003	analyzer.polito[1]	N2L6K2BN
URL	http://www.ethereal.com/icons/folder.gif	Thu Feb 22 06:46:04 1996	Tue Feb 4 12:47:27 2003	folder[1].gif	0PE341MV
URL	http://www.ethereal.com/icons/folder.gif	Thu Feb 22 06:46:04 1996	Tue Feb 4 12:47:27 2003	folder[1].gif	0PE341MV
URL	http://www.ethereal.com/icons/binary.gif	Thu Feb 22 06:45:53 1996	Tue Feb 4 12:47:27 2003	binary[1].gif	0PE341MV
REDR	http://s0b.bluestreak.com/ix.e?ir&s=114891&n=1036179023513505				
URL	http://www.ethereal.com/icons/binary.gif	Thu Feb 22 06:45:53 1996	Tue Feb 4 12:47:27 2003	binary[1].gif	0PE341MV
URL	http://www.ethereal.com/icons/back.gif	Thu Feb 22 06:45:53 1996	Tue Feb 4 12:47:27 2003	back[1].gif	0PE341MV
REDR	http://www.ethereal.com/distribution/win32/ethereal-setup-0.9.9.exe				
URL	http://analyzer.polito.it/	Fri Jan 17 03:57:15 2003	Tue Feb 4 12:48:38 2003	analyzer.polito[1]	N2L6K2BN
REDR	http://s0b.bluestreak.com/ix.e?ir&s=114890&n=1036179023513505				
LEAK	http://sfads.osdn.com/1.html		Tue Feb 4 11:56:29 2003	1[1].html	N2L6K2BN
URL	http://www.ethereal.com/icons/text.gif	Thu Feb 22 06:46:28 1996	Tue Feb 4 12:47:27 2003	text[1].gif	0PE341MV
URL	http://analyzer.polito.it/style.css	Tue Mar 20 09:46:47 2001	Tue Feb 4 12:50:00 2003	style[1].css	N2L6K2BN
LEAK	http://sfads.osdn.com/1.html		Tue Feb 4 11:56:29 2003	1[1].html	N2L6K2BN
URL	http://analyzer.polito.it/misc/help.htm	Wed Jan 9 09:14:51 2002	Tue Feb 4 12:49:46 2003	help[1].htm	N2L6K2BN
LEAK	http://prdownloads.sourceforge.net/ethereal/ethereal-setup-0.9.9.exe?use_default=easynews		Tue Feb 4 11:56:30 2003	ethereal-setup-0.9.9[1].html	N2L6K2BN
URL	http://analyzer.polito.it/cgi-bin/Count.cgi?df=analyzer.dat		Tue Feb 4 12:48:38 2003	Count[1].gif	N2L6K2BN
LEAK	http://prdownloads.sourceforge.net/ethereal/ethereal-setup-0.9.9.exe?use_default=easynews		Tue Feb 4 11:56:30 2003	ethereal-setup-0.9.9[1].html	N2L6K2BN
URL	http://analyzer.polito.it/images/capture.gif	Tue Mar 20 09:46:56 2001	Tue Feb 4 12:50:00 2003	capture[1].gif	0PE341MV
URL	http://analyzer.polito.it/cgi-bin/Count.cgi?df=analyzer.dat		Tue Feb 4 12:48:38 2003	Count[1].gif	N2L6K2BN
URL	http://analyzer.polito.it/style.css	Tue Mar 20 09:46:47 2001	Tue Feb 4 12:50:00 2003	style[1].css	N2L6K2BN
URL	http://analyzer.polito.it/install/default.htm	Sat Jan 11 08:52:34 2003	Tue Feb 4 12:50:00 2003	default[1].htm	N2L6K2BN
URL	http://analyzer.polito.it/IPv6-enabled.gif	Mon Sep 10 08:25:01 2001	Tue Feb 4 12:48:39 2003	IPv6-enabled[1].gif	0PE341MV
URL	http://www.ethereal.com/icons/text.gif	Thu Feb 22 06:46:28 1996	Tue Feb 4 12:47:27 2003	text[1].gif	0PE341MV
REDR	http://address.mail.yahoo.com/ya62/us/1648545519/82c63498				
URL	http://www.ethereal.com/icons/blank.gif	Thu Feb 22 06:45:54 1996	Tue Feb 4 12:47:27 2003	blank[1].gif	0PE341MV
URL	http://www.ethereal.com/icons/back.gif	Thu Feb 22 06:45:53 1996	Tue Feb 4 12:47:27 2003	back[1].gif	0PE341MV
REDR	http://www.verizonwireless.com/zip/plsql/vzw_zip.zip?p_alias=vzw_customer_care&p_url=&p_referer=VZW_ZIP_FORM&p_zip_code=22202&p_cookie=A				
URL	http://www.ethereal.com/distribution/win32/		Tue Feb 4 12:47:27 2003	win32[1].html	N2L6K2BN
REDR	http://www.verizonwireless.com/zip/plsql/vzw_zip.zip?p_alias=ec_contacts_b				
REDR	http://www.yahoo.com/r/m1				
URL	http://www.ethereal.com/icons/blank.gif	Thu Feb 22 06:45:54 1996	Tue Feb 4 12:47:27 2003	blank[1].gif	0PE341MV
REDR	http://s0b.bluestreak.com/ix.e?ir&s=114891&n=1036179023513505				
URL	http://www.ethereal.com/icons/binary.gif	Thu Feb 22 06:45:53 1996	Tue Feb 4 12:47:27 2003	binary[1].gif	0PE341MV
URL	http://www.ethereal.com/distribution/win32/		Tue Feb 4 12:47:27 2003	win32[1].html	N2L6K2BN
REDR	http://www.angelfire.com/doc/general/angelfire_popunder.html?search_string=%22ethereal%22+and+download				
REDR	http://data.coremetrics.com/eluminate?id%3D6%26ci%3D900083399%26vn2%3De3.1%26pc%3DY%26pi%3DHOME%2520PAGE%26jv%3D1.3%26je%3Dy%26ce%3Dtrue%26cp%3Dx86%26sw%3D1024%26s				
REDR	http://ad.doubleclick.net/ad/N2335.yahoo/B1072270.2;sz=468x60;ord=1036173795214003?				
URL	http://analyzer.polito.it/IPv6-enabled.gif	Mon Sep 10 08:25:01 2001	Tue Feb 4 12:48:39 2003	IPv6-enabled[1].gif	0PE341MV
URL	http://www.ethereal.com/distribution/win32/win32-README.txt	Mon Apr 15 23:01:34 2002	Tue Feb 4 12:47:33 2003	win32-README[1].txt	N2L6K2BN

Figure 23 - Pasco's Output

When running Pasco in the undeletion mode, it is possible that the numbers of rows are less when compared with the standard mode's output:

```
[kjones: pasco] kjones% ./pasco index.dat | wc -l
      53
[kjones: pasco] kjones% ./pasco -d index.dat | wc -l
      36
```

This phenomena is experienced when more than one activity record is inserted into the HASH table structure. If we are to sort out the unique activity records, we see that Pasco indeed returns more records when undeletion mode is enabled:

```
[kjones: pasco] kjones% ./pasco index.dat | sort -u | wc -l
      35
[kjones: pasco] kjones% ./pasco -d index.dat | sort -u | wc -l
      36
```

Pasco is open source and released under the liberal FreeBSD license.