

## Web Browser Forensics, Part 2

Keith J. Jones, Rohyt Belani 2005-05-11

### Reviewing part one

Welcome to part two of the Web Browser Forensics series. In [part one](#), we began investigating the intrusion of the Docustodian document management server hosting a law firm's data. The server appeared to have been compromised by a group of hackers who were using it as a repository for their MP3s, MPEGs, and pirated software.

In [part one](#), we also performed a review of the Internet Explorer history and cached files on the system used by Joe Schmo, the primary suspect of the intrusion. Analysis of the web browsing history revealed Internet searches for license cracks and hacking books; however, all this malicious activity appeared to have been performed while Joe was on vacation with his family in Florida.

In part two we now set out to determine who used Joe's machine while he was on vacation. We will proceed by examining further investigative leads that involve performing an in-depth review of the web activity of all other browsers installed on Joe's hard drive.

### The Investigation

Further investigation of Joe's system revealed that the only other web browser installed was Mozilla Firefox. Viewing Firefox cached files is not as straightforward as viewing IE cached files, simply because there is a lack of tools that provide an easy way to reconstruct the cache files. As we discussed in the last article, most tools like [FTK](#), [IE History](#), and [Web Historian](#), are able to reconstruct the history files for Mozilla based web browsers, but they do not associate the locally cached content to the web activity. This implies that we have the dates and times of web browsing activity, but cannot view the actual content of that activity, yet.

Let's begin the investigation by reviewing the Firefox cache on Joe Schmo's system. The directory is located at the following path:

```
\Documents and Settings\\Application Data  
\Mozilla\Firefox\Profiles\\Cache
```

Within the Cache directory, there are a number of files relevant to our investigation. Joe's cache directory is available below in Figure 1.

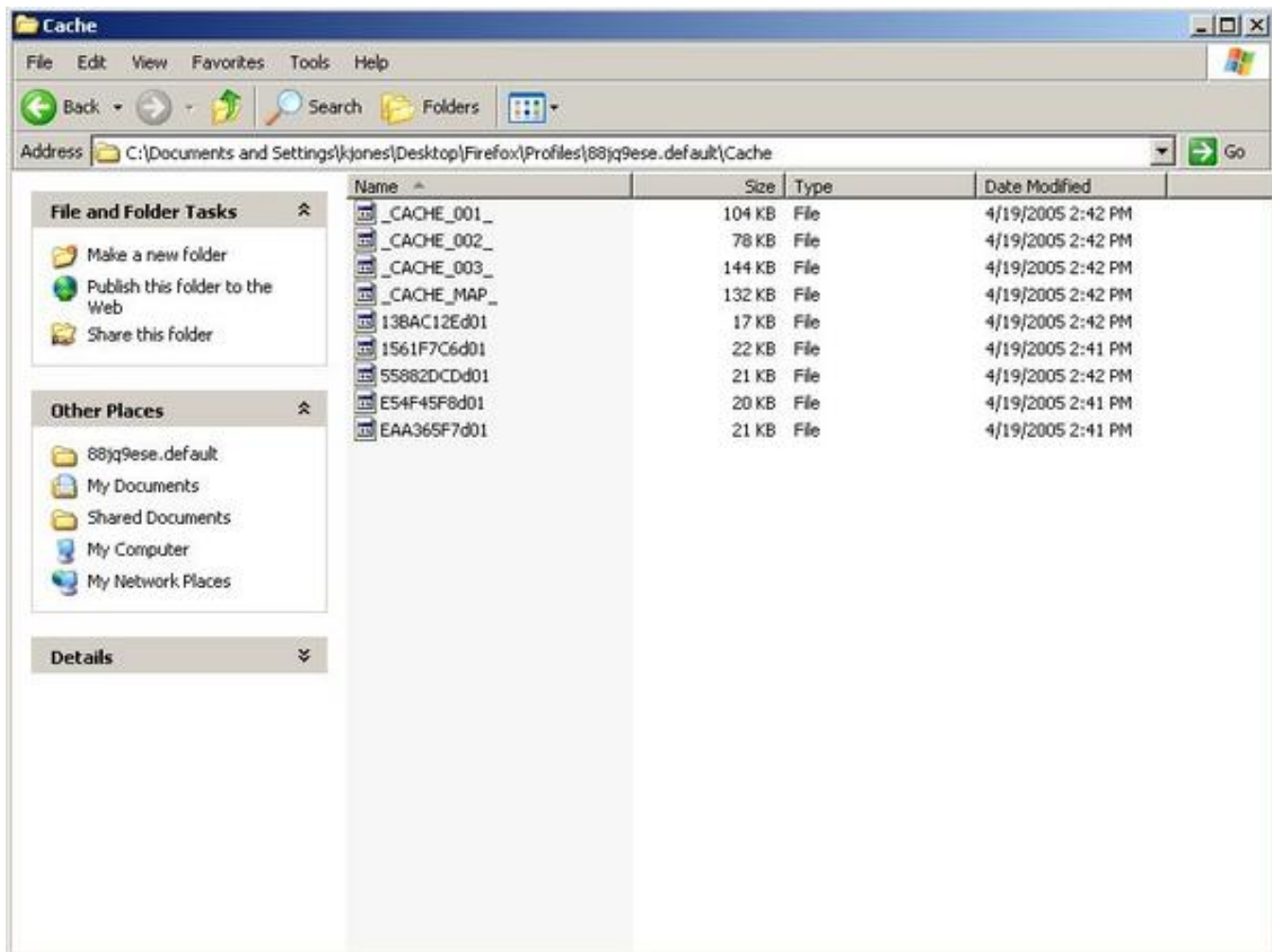


Figure 1: The Mozilla Firefox Cache Directory.

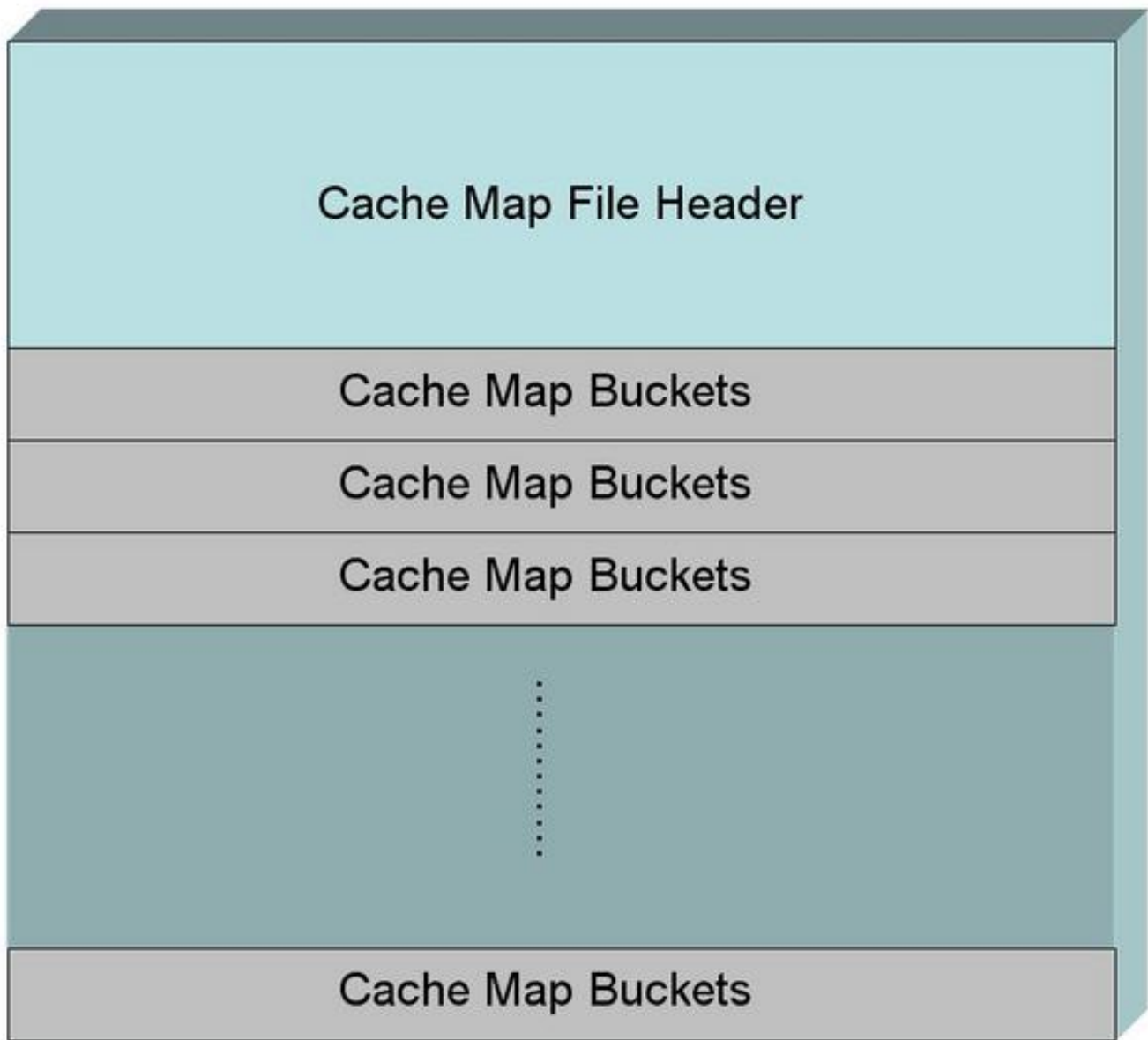
There are three types of files in this directory:

- A Cache Map File
- Three Cache Block Files
- Separate Cache Data Files

We will examine each of these files in order to reconstruct Firefox's cached data.

## Cache Map file

The Cache Map file is named "\_CACHE\_MAP\_" and is the main file used to reconstruct Firefox's cached activity. The Cache Map file has a very basic structure. There is a header for the file followed by "buckets" which contain the mapping information to the cached data, as illustrated in Figure 2.



**Figure 2: The Structure of `_CACHE_MAP_`.**

There are 32 buckets in this file. Within each bucket, there are 256 records inside each bucket. That means we have 8,192 records in the Cache Map file.

Each record contains the information for one instance of cache data. A record contains four 32-bit integers:

- A Hash Number
- An Eviction Rank
- The Data Location
- The Metadata Location

There are two methods Firefox uses to save the cache data. Firefox either saves the information inside a Cache Block File or creates a separate file. The hash number is used to name the separate file if that is how a specific cache instance was saved. The other two fields we will use to reconstruct the cache data are the "data location" and the "metadata location." Each instance of cache data has metadata information and the cache content.

If you take the "metadata location" field and "bitwise AND" it with 0x30000000 and then left shift the result 28 bits, you will have a number between zero and three. If the result is a zero, the cache metadata is saved in a separate file. If the result is one, two, or three, the cache metadata is embedded in a cache block file. The same algorithm using the data location as the input will identify similar information for the cache content.

## Cache Block files

There are three cache block files named "\_CACHE\_00N\_" where "N" is a number from one to three. Cache block files contain cache content and metadata information for each instance of cache activity. This structure of the cache block files presents a problem for most computer forensic tools attempting to reconstruct cache content during an investigation. However, we are aware of this structure and will extract the data in order to continue our investigation.

After the Cache Block File has been identified using the methodology presented above, we only need to know where the data is located in which appropriate Cache Block File. The start block is calculated by "bitwise ANDing" the metadata location/data location with 0x00FFFFFF. Next, we must calculate the number of contiguous blocks comprising the cache metadata/data. The size is calculated by "bitwise ANDing" the metadata location/data location with 0x03000000 and right shifting the result 24 bits. Now, we need to know how large the blocks are for a particular cache block file. Mozilla browsers define the block size by left shifting the number 256 by the following number of bits: subtract one from the cache block file number and then multiply the result by two. Therefore, when N=1 the block size is 256 bytes. When N=2, the block size is 512 bytes. When N=3, the block size is 1,024 bytes.

Lastly, we have to account for a bitmap header in the cache block file. The bitmap header is defined as 4,096 bytes long. The blocks in the cache block files begin immediately after the bitmap header. Using the algorithms presented above, we are able to extract the metadata and cache content for each record found in the cache map file.

## Separate Cache Data files

If the cache content or metadata is too large to be embedded in the Cache Block files, the information is saved into a separate Cache Data file. The file names are identified by the following format:

```
<HASH NUMBER><TYPE><GENERATION NUMBER>
```

The hash number is available from the Cache Map file. The "Type" is either 'd', for cache content, or 'm' for cache metadata. The "Generation Number" is an integer that is identified by "bitwise ANDing" the metadata location/data location with 0x000000FF.

## Reconstructing the Cache Data

The one tool that facilitates Firefox cache reconstruction is [Cache View](#). Cache View, a shareware tool, provides

access to the cached files of several types of browsers, including Firefox. For each cached page, Cache View provides the URL from which the page was retrieved, the name of the cached file as stored on the local system, the file size, file type, the time it was last modified, the download date and its expiry (if applicable). This information is the cache metadata that we discussed earlier. An example is shown below in Figure 3.

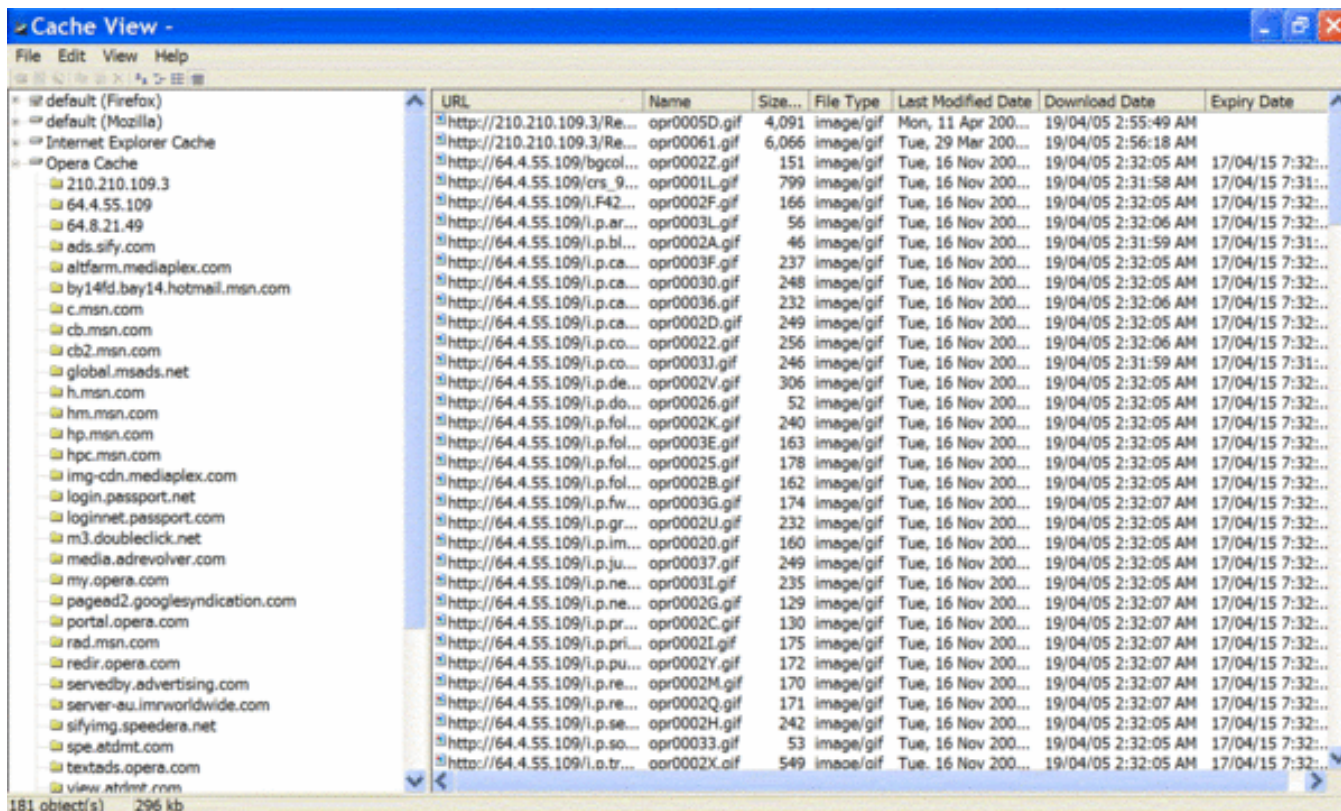
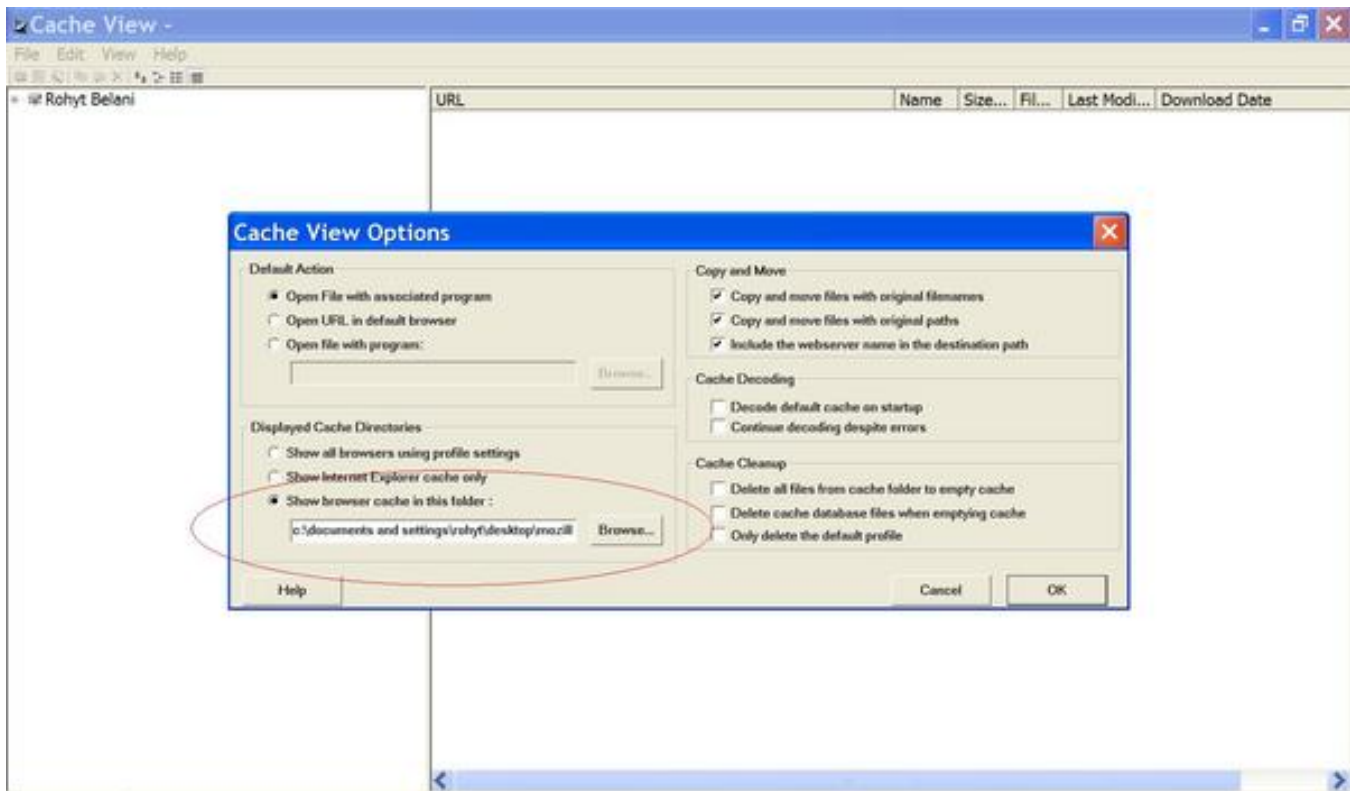


Figure 3: Presentation of browser cached files by Cache View.

By default, Cache View presents the files from the browser caches of the system on which it is running. In our case, as is the case with most forensics investigations, all investigative activity is performed on a forensics workstation, which is separate from the actual evidence media. Thus, we need to import the cached files of the browsers from a copy of the evidence medium onto the forensics workstation. In the case of Firefox, it means importing the following folder:

```
\Documents and Settings\\Application Data
\Mozilla\Firefox\Profiles\\Cache
```

Once the folder is stored at a known location on the forensics workstation, Cache View can be instructed to retrieve files from it, rather than the browser caches of the forensics workstation. This is shown below in Figure 4.



**Figure 4: Pointing Cache View to a copy of the Firefox cache folder of the evidence medium, that is resident on the forensic workstation.**

A zipped version of the Firefox Cache folder from Joe Schmo's system, used for example purposes in this article, can be downloaded from this link.

After importing the cache folder into Cache View you should be presented with a view similar to that shown in Figure 5, below.

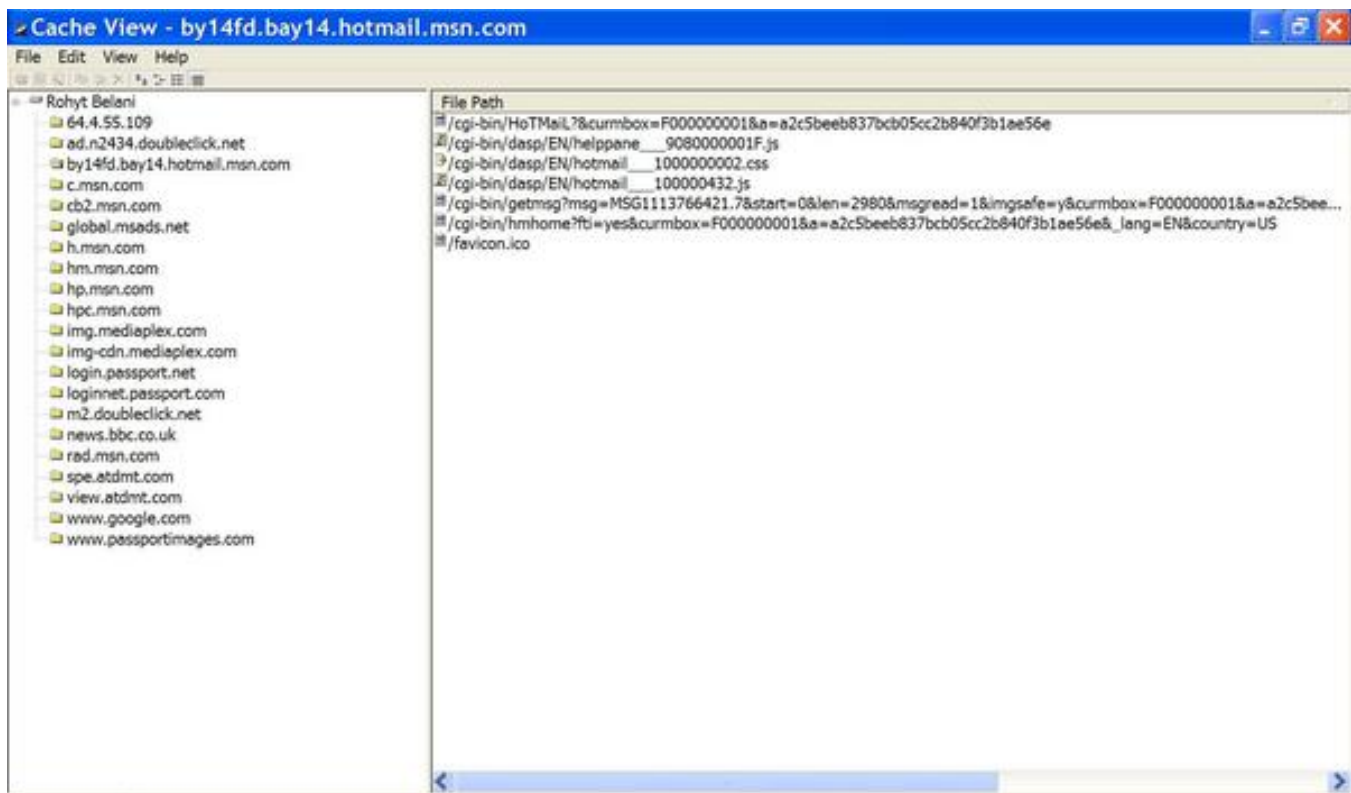


Figure 5: View of the cached files from the Firefox cache on Joe Schmo's system.

As seen in Figure 5, the files are categorized by the domain names from which they were retrieved. By clicking the `by14fd.bay14.hotmail.msn.com` folder in the left pane of Cache View, we observe a number of links to visited pages within this domain in the right pane of Cache View. These web pages may provide insight into additional Hotmail activity recorded on Joe Schmo's machine -- this piques our interest! We decide to embark on a journey to view these files. This process entails the following:

1. **Copying the visited web pages into a folder at a known location (e.g. Desktop)**

This operation is achieved by selecting all the links in the right-pane, right-clicking the mouse and selecting the Copy to.. option as shown below in Figure 6.

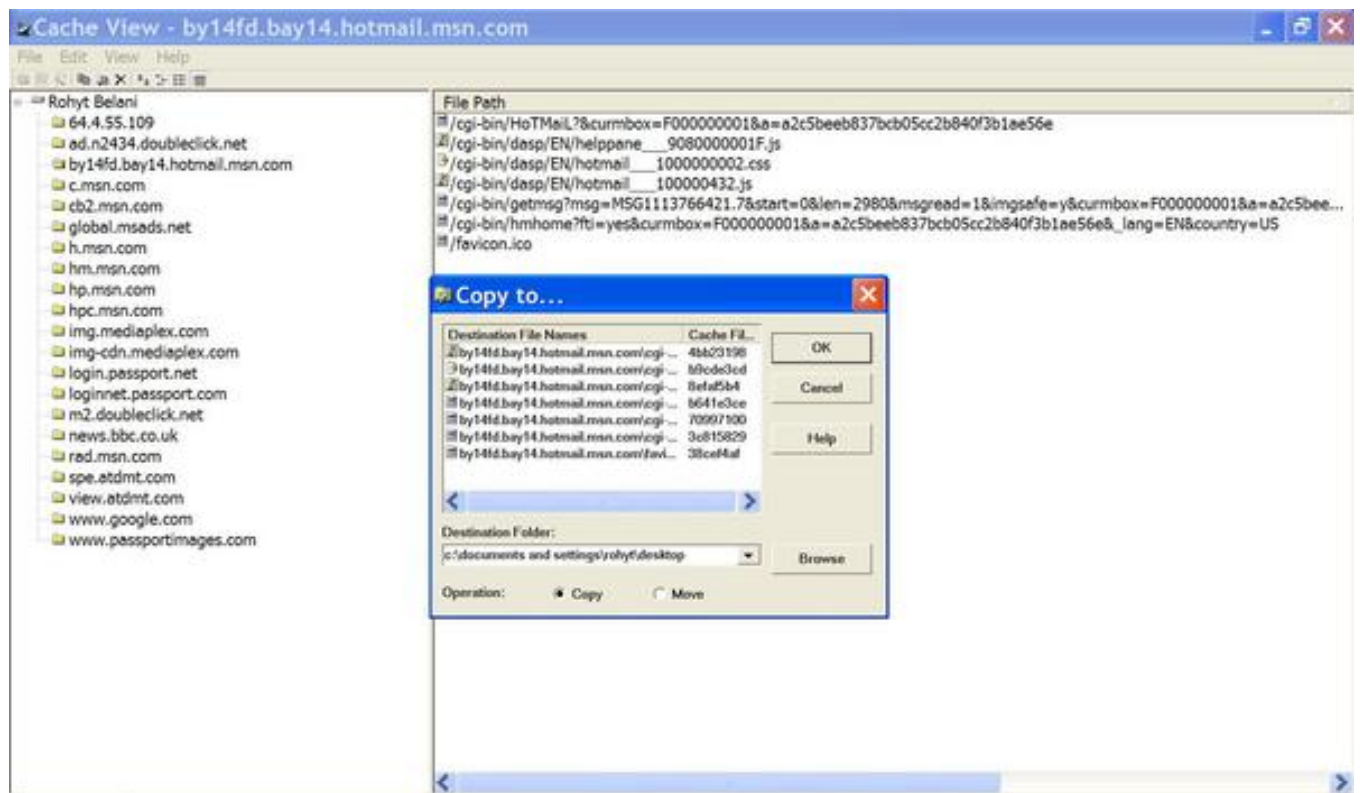


Figure 6: Copying the cached web pages from Cache View to the Desktop of the forensics workstation.

## 2. Unzipping the copied files

Following step 1, the files were saved in a folder called `by14fd.bay14.hotmail.msn.com` on the forensics workstation's desktop. We then ran the Unix "file" command (in a Cygwin environment) against all the files in this folder.

```

rohyt@homer /cygdrive/c/Documents and Settings/rohyt/Desktop/by14fd.bay14.hotmai
l.msn.com/cgi-bin
$ file *
HoTMaiL_&curmbox=F000000001&a=a2c5beeb837bcb05cc2b840f3b1ae56e:
                                     gzip compressed data, from MS-DOS, max sp
eed
dasp:
                                     directory
getmsg_msg=MSG1113766421.7&start=0&len=2980&msgread=1&imgsafe=y&curmbox=F0000000
01&a=a2c5beeb837bcb05cc2b840f3b1ae56e: gzip compressed data, from MS-DOS, max sp
eed
hmhome_fti=yes&curmbox=F000000001&a=a2c5beeb837bcb05cc2b840f3b1ae56e&_lang=EN&co
untry=US:
                                     gzip compressed data, from MS-DOS, max sp
eed
rohyt@homer /cygdrive/c/Documents and Settings/rohyt/Desktop/by14fd.bay14.hotmai
l.msn.com/cgi-bin
$ =

```

Figure 7: Output of the "file" command.

Figure 7 indicates that the files are gzip compressed. In order to decompress the files, we created copies of the files, named them 1.gz, 2.gz and 3.gz respectively, and then ran the "gunzip" utility against them.

```

rohyt@homer /cygdrive/c/Documents and Settings/rohyt/Desktop/by14fd.bay14.hotmai
l.msn.com/cgi-bin
$ ls
1.gz
HoTMaiL_&curmbox=F00000001&a=a2c5beeb837bcb05cc2b840f3b1ae56e
dasp
getmsg_msg=MSG1113766421.7&start=0&len=2980&msgread=1&imgsafe=y&curmbox=F0000000
01&a=a2c5beeb837bcb05cc2b840f3b1ae56e
hmhome_fti=yes&curmbox=F000000001&a=a2c5beeb837bcb05cc2b840f3b1ae56e&_lang=EN&co
untry=US

rohyt@homer /cygdrive/c/Documents and Settings/rohyt/Desktop/by14fd.bay14.hotmai
l.msn.com/cgi-bin
$ gunzip -d 1.gz

rohyt@homer /cygdrive/c/Documents and Settings/rohyt/Desktop/by14fd.bay14.hotmai
l.msn.com/cgi-bin
$ =

```

Figure 8: "Gunzip"ing 1.gz the renamed version of the getmsg[...] file.

### 3. Opening the unzipped file using Firefox

We then proceeded to opening 1.gz using Firefox. The following screenshot, Figure 9, shows the actual visited page that was re-constructed by this process.

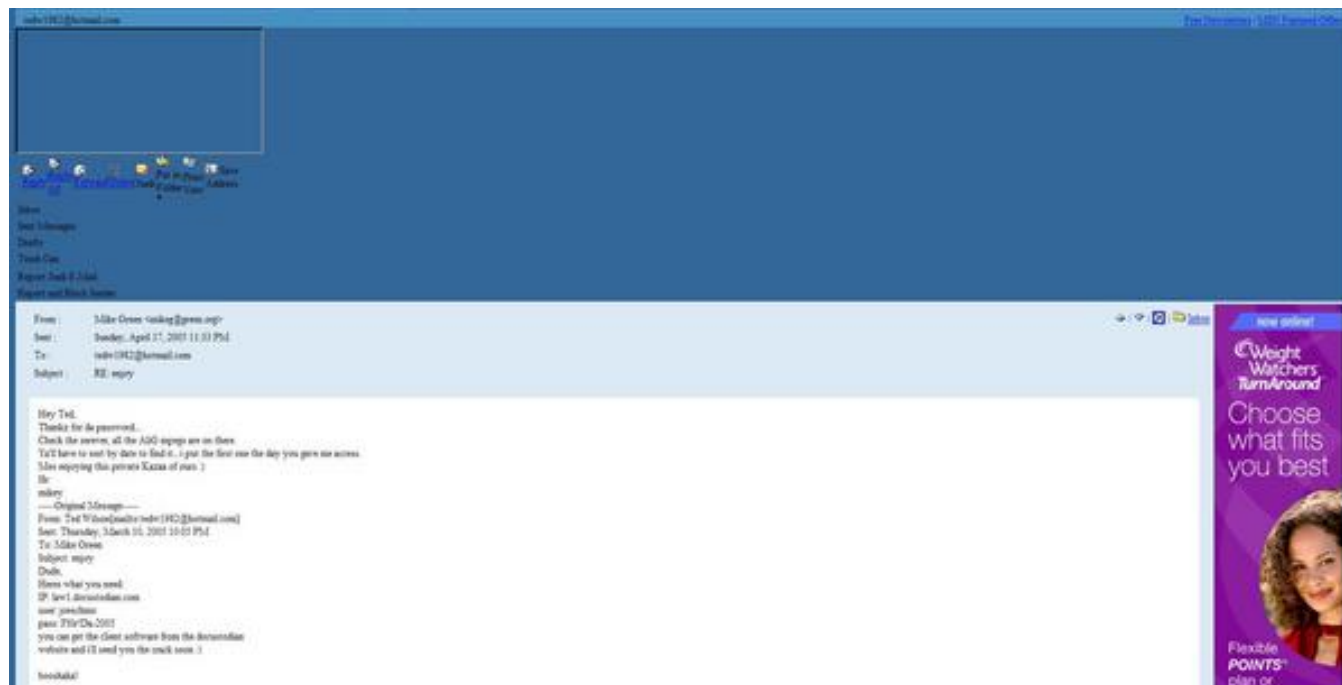


Figure 9: Reconstructed Firefox web activity.

The extracted web page shown in Figure 9, displays an email viewed on Joe Schmo's machine. Careful examination of the page reveals that the e-mail resides in the Hotmail inbox of tedw1982@hotmail.com.

The content of the email is extremely pertinent to this case. It indicates that Ted Wilson, the owner of the tedw1982@hotmail.com email account, sent Mike Green an email providing him Joe Schmo's credentials for the victim Docustodian server, discussed in part one. Ted also indicated in the email to Mike Green that the client software for Docustodian may need a license crack that he would send shortly. Furthermore, the email was sent on March 10, 2005 at 10:05PM, while Joe was on vacation with his family. So the cat is now out of the bag! Ted Wilson is guilty of master-minding the unacceptable use of the Docustodian server.

So who is Ted Wilson and how did he gain access to Joe Schmo's system? Conversations with personnel at the law firm indicated that Ted was the intern employed to cover for Joe Schmo when he was on vacation.

## The last nail in the coffin

If the extracted Hotmail message wasn't evidence enough, perusal of the evidence media revealed a file called `licensecrack.java` in the following directory:

```
C:\windows\system32\temp\temp\temp\
```

This file presented a last access time stamp of 07:32PM on March 11, 2005. An excerpt from `licensecrack.java` is listed below:

```
/*
 * This program should be run on the same LAN
 * as the Docustodian client machine.
 * Modify the hosts file on the client machine accordingly.
 * It tricks the client in believing that it has a valid license
 * to access the server
 * Author: Ted Wilson
 */

import java.net.*;
import java.io.*;
import java.io.ByteArrayOutputStream;

public class License
{
    static DatagramSocket    sock;           // socket
    static DatagramPacket    opkt;          // communications packet
    static DatagramPacket    ipkt;          // communications packet
    static InetAddress       clientIP;      // IP address of
client
```



## About the authors

[Keith J. Jones](#) is Director of Computer Forensics and Incident Response at [Red Cliff Consulting](#).

[Rohyt Belani](#) is Director of Proactive Security Services at [Red Cliff Consulting](#).

View more articles by [Rohyt Belani](#) on SecurityFocus.

[Privacy Statement](#)

Copyright 2006, SecurityFocus