Forensic Analysis of Microsoft Internet Explorer Cookie Files

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

5/1/03
(revised 5/6/03)

Table of Contents

Listing of Tables

1. Introduction

Since HTTP is a stateless protocol, websites must place information on a user's computer if it needs to save information about a web session. For instance, when a user selects a widget and adds it to his shopping cart, that information can be saved on the client computer rather than the web server. The facility to save information in this manner is known as *Cookies*. A cookie is a small file containing data that the web server places on a user's computer so it may request back at a later date.

During forensic analysis it is often relevant to parse the information in Internet Explorer's cookie files into a human readable format. Cookies aid forensic analysts during the investigation by providing insight to a suspect's internet activity. After analysis of several example cookie files it was found that the format is relatively simple to understand. This paper will document the format of Internet Explorer's (IE) cookie files for forensic analysis purposes and provide an open source tool to parse the information into a human readable format.

2. The IE Cookie File Format

After visiting a website such as *www.securityfocus.com*, a cookie will be generated on the user's computer that looks similar to the following:

```
sffocus
home
securityfocus.com/
0
1238799232
29570658
1484443312
29552553
*
```

This cookie contains the information meant to be saved on the client from the web server, the domain name that is responsible for this cookie, and the relevant time/date stamps. The file will be created in the user's IE Cookie directory, typically located in the following places:

Table 1 - Common IE Cookie File Locations

Operating System	*Cookie File Location*
Windows 2000/XP	`C:\Documents and Settings\<username>\Cookies`

Windows 95/98/ME	C:\Windows\Cookies

The file will be named for website the cookie belongs to, followed by a `.TXT` file extension. The cookie in this example was located at `C:\Documents and Settings\Administrator\Cookies\administrator@securityfocus[1].txt` on a Windows 2000 machine.

Notice the file is in ASCII format, so a human is able to easily read the contents. We will analyze this file line by line. The first line contains the variable name. In this case, the variable is named `sffocus`. The second line contains the value for the variable. In this example, the variable `sffocus` has the value of `home`. The third line contains the website that issued the cookie. In this example, the website is `securityfocus.com`. The fourth line contains flags, which are zero in this case.

The next two lines (lines five and six) contain the expiration time for the cookie. This is the time when the cookie will not longer be valid for `securityfocus.com`. The time is separated into 2 integers. The first integer is the most significant integer of the expiration time. The second is the least significant integer. The time may be reassembled by concatenating the most significant value (line five) to the least significant value (line 6). For example, if the most significant value was 1 and the least significant value was 2, the expiration time would be (in binary):

```
00000000000000000000000000000001 00000000000000000000000000000010
```

Once the time has been reassembled in this manner, it is evident that it is saved in Microsoft Windows's FILETIME format. FILETIME format is the number of ticks, in 100 nanosecond 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 next two lines (lines 7 and 8) are the creation time for the cookie. The creation time must be reassembled in the same manner as the expiration time. The last line (line 9) will always contain a * since it is the record delimiter when this text file contains more than one cookie. A new cookie would start on the next line (line 10).

The following table summarizes the lines in the cookie file:

Table 2 - The IE Cookie File Format Summary

Line	*Summary*
1	The Variable Name
2	The Value for the Variable
3	The Website of the Cookie's Owner
4	Optional Flags
5	The Most Significant Integer for Expired Time, in FILETIME Format
6	The Least Significant Integer for Expired Time, in FILETIME Format
7	The Most Significant Integer for Creation Time, in FILETIME Format
8	The Least Significant Integer for Creation Time, in FILETIME Format
9	The Cookie Record Delimiter (a * character)

3. Galleta – The Open Source IE Cookie File Parser

Now that we understand the internal structures of a cookie file, we can develop a tool to automate everything we have done so far. The author developed a tool named Galleta, the Spanish word for cookie, to parse the information in a cookie file and return the results in a field delimited format. Galleta's command line arguments are as follows:

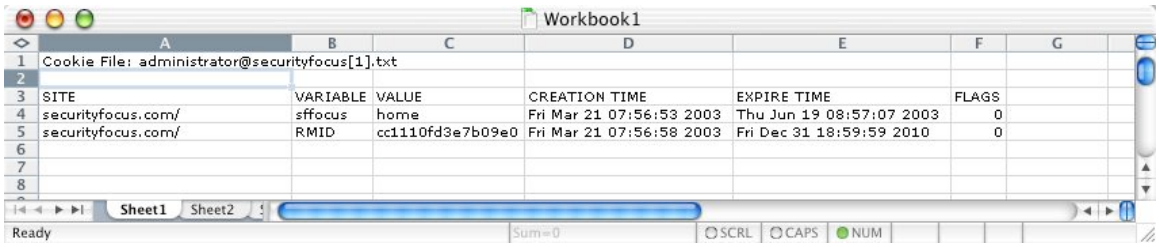
```
[kjones: galleta] kjones% ./galleta

Usage:  galleta [options] <filename>
        -t Field Delimiter (TAB by default)
```

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 Galleta is run in the following manner:

```
./galleta administrator@securityfocus[1].txt > securityfocus[1]_galleta.txt
```

Galleta shows us that this cookie file contains two cookies from `securityfocus.com`. It is important to note that Galleta's output can be easily imported into your favorite spreadsheet program so that you may sort, search, and filter the data. Furthermore, a spreadsheet will allow you to format the data so that it is appropriate for a report.



Galleta is open source and released under the liberal FreeBSD license. Galleta can be compiled on Windows (using Cygwin), Mac OS X, Linux, and *BSD.