Running Sleuthkit and Autopsy Under Windows

Published June 11, 2004
Charles Lucas
charles@lucas.net
http://www.memophage.net

This tutorial is intended for people who are familiar with Windows and Windows-based forensics tools, but may not be familiar with UNIX-style syntax, or the Cygwin, Sleuthkit, or Autopsy toolsets. Note that some of the commands herein assume an NT-kernel based system (Windows NT, 2K, XP), and may not function correctly on a Windows 9X or Millennium Edition system.

Cygwin is a Linux-like environment for Windows. It essentially pretends to be a Linux system, but intercepts the Linux system calls and routes them to the requisite Windows system calls instead. It comes with the gcc compiler, so it is often possible to recompile Linux-based applications to run under Cygwin on a Windows environment. In the case of Sleuthkit and Autopsy, the developers intentionally made sure that these utilities would compile under Cygwin, making our task much easier.

Autopsy is a web-based graphical interface that uses the Sleuthkit command-line tools. Thus, we must first install the Cygwin environment, the compile the Sleuthkit tools under that environment. Once that is complete, we can install Autopsy, which will start its own local web server application that we can use to orchestrate the Sleuthkit tools.

Installing Cygwin

Browse to www.cygwin.com, and click on one of the install links with the icon to download setup.exe.

When the File Download dialog window pops up, hit “Open”, to run the executable.

This should download and start the “Cygwin Net Release Setup Program”. Click “Next”, check that “Install from Internet” is selected, and hit “Next” again.

Set your Cygwin Root directory. Usually I set it to the root of a drive, such as C:\cygwin, or D:\cygwin. Keeping defaults for the other settings are fine. *From here on, whenever you see C:\cygwin, substitute your own root directory.*

Set the location of the package directory. I usually set it to be under the root Cygwin directory, such as C:\cygwin\package. This directory will hold the software packages as they are downloaded. Click “Next”.

Make sure “Direct Connection” is selected, and click “Next” again. The setup program should download a list of mirror sites. Which one is best depends a lot on your location, but I’ve found ftp://mirrors.kernel.org to be consistent and fast. Highlight the chosen server, and click “Next”.

The Setup program will download a list of all available packages and their current versions. It helps if you grab the bottom-left corner of the window and expand it to see more fields. By default, the setup program will install the basic Cygwin packages, but we need some additional utilities to build Sleuthkit and Autopsy.

Click on the word “Devel”. This should expand the development package tree. Scroll down until you see “gcc: C compiler”. Click on the word “skip” on that line: it should become a version number. As of the time of this writing, gcc is at version 3.3.1-3. Follow these steps for this list of additional packages:

Under “Devel”:

gcc:	C compiler	(completed)
gcc=g++:	GCC C++ compiler	
make:	The GNU version of the ‘make’ utility	
patchutils:	A small collection of programs that operate on patch files.	

Under “Interpreters”:

perl:	Larry Wall’s Practical Extracting and Report Language

When all desired packages are selected, click “Next”. Setup will download all the packages and install them. When it is finished, click “Finish” to create desktop and Start Menu Icons.

Compiling Sleuthkit

Sleuthkit is a collection of command-line utilities which are written in C. No Cygwin-specific binaries are currently available, so we have to download the source code and compile it ourselves:

Browse to http://www.sleuthkit.org/sleuthkit

Click on the “Download” link in the left column, and then the “Source Code” link under the current version.

Click on the graphic next to the mirror you want to download from (Minneapolis seems to work well), and the download should start in a few seconds. As of this writing, the filename is sleuthkit-1.70.tar.gz. *From now on, whenever you see “sleuthkit-1.70”, substitute the version you downloaded.*

When the “File Download” dialog window pops up, click “Save”, and then move to Cygwin’s /usr/local directory. (C:\cygwin\usr\local). Click “Save” to download the file into that directory.

You should have a Cygwin icon on your desktop with the graphic. Double-click it to open a Cygwin Bash shell.

Note: Cygwin should automatically create a user directory under C:\cygwin/home with the name of the user you are currently logged in as. Enter the ‘pwd’ command to see what your home directory is.

Change to /usr/local by entering ‘cd /usr/local’. Uncompress the Sleuthkit archive file by entering the command ‘tar xvfz sleuthkit-1.70.tar.gz’. Cygwin supports tab-completion, so you should be able to type the first few letters of the filename, then hit the tab key to have the rest filled in).

Change into Sleuthkit’s directory with ‘cd sleuthkit-1.70’.

Enter the command ‘make’. The compilation process should begin.

In versions 1.70 and previous, there is a bug in the magic.c source file. If the bug still exists in the version you downloaded, the build process will fail with a message like "magic.c:192: error: storage size of 'utsbuf' isn't known.

Ian Lance Taylor created a fix for this, which I found in a NetBSD newsgroup archive at http://news.gw.com/netbsd.tech.toolchain/4094. I have copied this e-mail into Appendix A of this document for convenience.

You will need to copy the block of code from the document from (inclusive) the line "Index: magic.c" to (inclusive) "#ifdef HAVE_UNISTD_H". Open up notepad and paste that block. Save the document in the directory C:\cygwin/usr/local/<sleuthkit>/src/file/src as "patch.txt".

Switch back to the Bash shell, and change into the source directory that contains magic.c with the command 'cd src/file/src'. Make a backup copy of magic.c by running the command 'cp magic.c magic.bak'. Merge the patch into the original magic.c with the command 'patch magic.c patch.txt'. If everything goes right, the command should output a message like "Hunk #1 succeeded at 42 (offset -2 lines)".

Change back to the <sleuthkit> directory with 'cd ../../..', and run the command 'make' again. This should compile the rest of the program and complete with the lines "Checking Tools", and "Done".

Move back to /usr/local/ with 'cd ..', and make a "soft link" to the Sleuthkit-1.70 directory with the command 'ln –s /usr/local/sleuthkit-1.70 sleuthkit'. In this way, you can install new versions of Sleuthkit, and change this shortcut to point to it, without having to reconfigure Autopsy or your system path.

You now have Sleuthkit compiled, and you can run the command-line tools by change into Sleuthkit's bin directory ('cd sleuthkit/bin'). You will have to execute them with the syntax './<file>' until you add the full bin directory path to your system's PATH environment variable. A good description of all the Sleuthkit utilities is at http://www.sleuthkit.org/sleuthkit/tools.php.

Accomplish the PATH modification by opening Windows' Control Panel, double-clicking System, clicking on the "Advanced" tab, and then on the "Environment Variables" button. In the lower panel, scroll down until you see "path". Click on that line, and click "Edit". In the new window, hit right-arrow until you're at the end of the line, add a semicolon (;) if there isn't one already, and add Sleuthkit's bin directory ("C:\cygwin\usr\local\sleuthkit\bin"). *Note the back-slashes. It can be confusing since Cygwin uses forward-slashes and Windows uses back-slashes to denote sub-directories.*

Switch back to the Bash window and enter the command 'exit' to close it down. It is necessary to open a new one to propagate the new PATH modifications.

One last thing: when using Autopsy, some of the Sleuthkit files will have trouble finding the Cygwin system DLL since they're not being executed from the command line. The easy fix for this is to make a copy of the DLL in the same directory as the Sleuthkit command-line binaries. Go back to the Bash shell and enter the command 'cp /bin/cygwin1.dll /usr/local/sleuthkit/bin' to copy the DLL.

Installing Autopsy

Autopsy is a web-based front end for the Sleuthkit command-line tools, which is written in the PHP scripting language. As such, it doesn't require compiling, but we need to download and decompress it, and then set configurations so it can find the Sleuthkit tools.

Browse to http://www.sleuthkit.org/autopsy.

Click on the "Download" link in the left column, and then the "Source Code" link under the current version.

Click on the graphic next to the mirror you want to download from (Minneapolis seems to work well), and the download should start in a few seconds. As of this writing, the filename is autopsy-2.01.tar.gz. *From now on, whenever you see "autopsy-2.01", substitute the version you downloaded.*

When the "File Download" dialog window pops up, click "Save", and again move to Cygwin's /usr/local directory, if you're not still there from last time. (C:\cygwin\usr\local). Click "Save" to download the file into that directory.

Open a new Bash shell with the Cygwin icon on your desktop. Change back to C:/cygwin/usr/local ('cd /usr/local'). Uncompress Autopsy by entering the command 'tar xvfz autopsy-2.01.tar.gz'.

Make a "soft link" to autopsy-2.01 directory with the command 'ln –s /usr/local/autopsy-2.01 autopsy'. In this way, you can install new versions of Autopsy, and change this shortcut to point to it, without having to reconfigure other tools or your system path.

You will also need to make an "evidence" directory with the command 'mkdir evidence'.

Change into Autopsy's directory ('cd /autopsy-2.01'), and enter the 'make' command to begin the configuration process.

The configuration script should automatically detect the locations for PERL and the 'grep' utility. It will ask you for the location of the Sleuthkit tools. Type "/usr/local/sleuthkit", and hit Enter. For some reason, sometimes the script can't find it the first time, and you have to enter it again.

Next, it will ask you "Have you purchased or downloaded a copy of the NSRL (y/n) [n]". The NSRL is a "known file" list published by NIST, and is available at http://www.nsrl.nist.gov. The NSRL consists of four CDs worth of filenames, sizes, and hashes for known good system files, so that developers can write utilities (like Sleuthkit) that can detect whether a system file has been modified or not. It seems that the best way to handle these is to download the ISO files, and then find a utility that will pretend they are actual CDs and mount them with drive letters. Installing these is recommended, but optional, so for now just press "Enter" for "no". If you install them later, you'll have to re-run Autopsy's configuration script.

The script will then ask you for the location of the "evidence locker directory". Enter "/usr/local/evidence".

The script should complete with the statement "Execute the './autopsy' command to start with default settings. Enter the command './autopsy', and the Autopsy server should begin execution. It starts its own http service on port 9999.

You should see the message “Open an HTML browser and on the remote host and paste this URL in it: http://localhost:9999/autopsy.

Pull up a browser window and past this URL into the address field. This will direct the browser to Autopsy’s http server on your local machine. You should see the “Sleuthdog” graphic, and options to open a case, start a new case, or access the Help menu. Congratulations, you’re now running Autopsy.

At the Autopsy Home Page (http://www.sleuthkit.org/autopsy/) are links to documentation as well as the Informer, a “bi-monthly newsletter for The Sleuth Kit, Autopsy, and related tools”.

Creating a Disk Image

Now that we have Sleuthkit and Autopsy installed, we need something to analyze. There are web sites that have disk images available for tool testing (such as http://dftt.sourceforge.net/), but we’ll walk through making an image of a floppy disk with Cygwin’s version of the dd utility.

First, find a floppy disk with something interesting on it, and put it in your floppy drive. Open a new Bash shell with the Cygwin icon on your desktop. Change to Cygwin’s /usr/local directory (‘cd /usr/local’). Make a new directory called “images” (‘mkdir images’), and change into it (‘cd images’).

Execute the dd command, specifying the a: drive as the source “input file”, and “floppy1.img” as the “output file”. The command will look like ‘dd if=//./a: of=floppy1.img’\

dd will make a bit-for-bit copy of the floppy disk into floppy1.img, and will exit with a statement like:

```
2880+0 records in
2880+0 records out
```

If you wanted to make an image of one partition on a drive (in this example, the d: drive) you would use the syntax like:

```
dd if=/cygdrive/d of=ddrive1.img
```

If you wanted to make an image of an entire physical hard drive (in this example, the second drive in the system), you would use syntax like:

```
dd if=//./physicaldrive1 of=physicaldrive1.img
```

Note: the //./ syntax denotes direct I/O access to the device in question. You need a write-blocker on the subject hard drive to take forensically sound drive images.

Switch back to your Autopsy browser window. When you create a new case, add a new host, and then add evidence to the case, point Autopsy to /usr/local/images/floppy1.img, and tell it to copy the image into the evidence locker. You will also need to tell Autopsy the file system type, which is fat12, and the original mount point, which would be //./a:

If you get an error message about “The dynamic link library cygwin1.dll could not be found”, go back up to the end of the “Compiling Sleuthkit” section and make sure you copied the cygwin1.dll file into the /usr/local/sleuthkit/bin directory.

Appendix A

(from http://news.gw.com/netbsd.tech.toolchain/4094)

Minor buglet in file causes cygwin build failure

Subject: Minor buglet in file causes cygwin build failure
From: ian [at] wasabisystems.com (Ian Lance Taylor)
Newsgroups: netbsd.tech.toolchain
Organization: TAC News Gateway
Date: May 21 2004 04:55:03

The file src/dist/file/src/magic.c can use either utime or utimes. The choice is made in one function, close_and_restore(). If HAVE_UTIMES is defined, that function uses utimes. Otherwise, if HAVE_UTIME_H or HAVE_SYS_UTIME_H is defined, that function uses utime.

That is fine. However, at the top of the file, when including header files, it checks for HAVE_UTIME before checking for HAVE_UTIMES. In particular, <sys/time.h> is only included if HAVE_UTIME is not defined.

When cross-building the tools on cygwin (please don't ask), both HAVE_UTIME and HAVE_UTIMES are defined, so <sys/time.h> is not included. The compilation then gets an error because struct timeval is not defined:

```
/home/ian/wasabisrc/src/dist/file/src/magic.c: In function `close_and_restore':
/home/ian/wasabisrc/src/dist/file/src/magic.c:198: error: storage size of `utsb\uf'
isn't known
```

I have appended what seems to me to be the obvious patch: to check for HAVE_UTIMES before HAVE_UTIME when including header files, so that the header file inclusion matches the usage.

(By the way, in the HAVE_UTIMES case, the tv_usec fields of the struct timeval arguments to utimes are uninitialized. I suppose this doesn't matter because current file systems will ignore the microsecond values anyhow. But it seems like a lurking bug. I haven't included the obvious fix.)

Thanks.
Ian

```
Index: magic.c
===================================================================
RCS file: /cvsroot/wasabisrc/src/dist/file/src/magic.c,v
retrieving revision 1.1.1.5
diff -p -u -r1.1.1.5 magic.c
--- magic.c    10 May 2004 04:18:24 -0000    1.1.1.5
+++ magic.c    21 May 2004 04:36:05 -0000
@@ -44,14 +44,14 @@
 #include <sys/mman.h>
 #endif

-#if defined(HAVE_UTIME)
+#if defined(HAVE_UTIMES)
+# include <sys/time.h>
+#elif defined(HAVE_UTIME)
```

```
 # if defined(HAVE_SYS_UTIME_H)
 #  include <sys/utime.h>
 # elif defined(HAVE_UTIME_H)
 #  include <utime.h>
 # endif
-#elif defined(HAVE_UTIMES)
-# include <sys/time.h>
 #endif

 #ifdef HAVE_UNISTD_H
```

---

www [at] gw.com