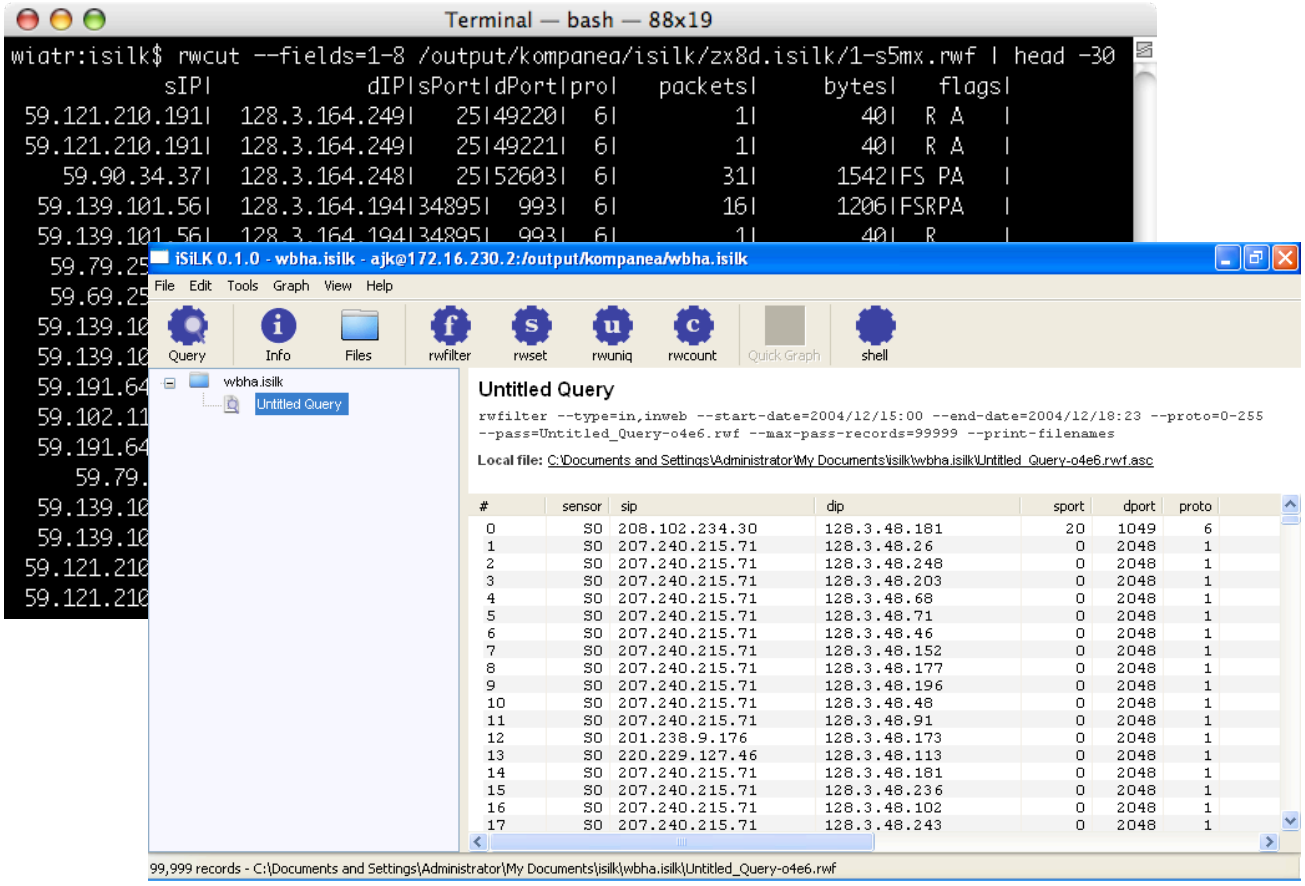
iSiLK

A graphical front-end for the SiLK tools

Development & Deployment Guide

for iSiLK version 0.1.2

November 2008



iSiLK was developed by the

Network Situational Awareness Group at CERT
Software Engineering Institute
Carnegie Mellon University

iSiLK User Guide

Table of Contents

Table of Contents 3

Installing iSilk from Source 5

Introduction 5

Installation Steps 5

Step 1 – Install iSiLK Dependencies 5

Step 2 – Install Python Source Code 7

Step 3 – Test your SSH connection 7

Step 4 - Run iSiLK with Python 7

Building a Windows Binary Distribution 9

Introduction 9

Building the Windows Installer 9

Deployment Considerations 11

Where iSiLK Stores Files 11

Local Configuration Files 11

Local Data Files 11

Remote Data Files 11

Installing iSilk from Source

Introduction

This section includes information about running iSILK source directly from a Python interpreter and for building a binary distribution from the source code. If you'd like to do iSiLK development, including writing a plug-in, you'll need to set up a development environment following the steps in this guide. If all you need to do is install iSiLK on your Windows desktop, see the iSiLK **User's Guide**.

Although the instructions in this section are geared toward Windows, it should be fairly straightforward to translate these instructions into the appropriate steps on a Linux system or Mac OSX. iSiLK should run anywhere the Python packages it depends on are available. The graphical user interface is based on wxPython, which is in turn based on wxwidgets, a platform-independent graphical user interface library that can be used to build applications on Linux under a variety of windowing environments, under Mac OSX and of course, under Microsoft Windows.

Installation Steps

Step 1 – Install iSiLK Dependencies

In order to run iSILK you'll to install all of the following open source software.

1. **Install Python 2.4**
   iSiLK is an application written in Python. The iSiLK sources should run in a version 2.4 Python interpreter or newer, although it has only be tested under Python 2.4. It has been tested on Windows with both ActivePython, a version of Python maintained by ActiveState, a commercial software company that offers various support packages, and with the version available from Python.org. They may be downloaded from http://www.activestate.com/ and http://www.python.org respectively. If you install the Python.org distribution on Windows, you will also need to add C:\Python24 to your executable path. To do this, go to the Windows Control Panel, Advanced Tab and select the Environment Variables button. Select the *Path* System variable, choose Edit, and add C:\Python24 at the end making sure to put a semicolon between it and any preceding Path variables.
2. **Install additional Python modules**
   Be sure to install the variants of modules intended for a Python 2.4 interpreter.. The module version numbers below correspond to versions of the modules that have been used in testing. Later versions may work but these have not been tested.

   wxPython 2.8 (user interface library)

http://www.wxpython.org/download.php
wxPython2.8-win32-unicode-2.8.4.0-py24.exe

NumPy 1.2.0 or higher (required by MatPlotLib)

http://sourceforge.net/project/showfiles.php?group_id=1369&package_id=175103

MatPlotLib 0.98.3 or higher (graphing)

http://sourceforge.net/project/showfiles.php?group_id=80706

In order to enable built-in ssh support, provide the following:

Paramiko 1.7.2 (client ssh library)

http://www.lag.net/paramiko/download/paramiko-1.7.2.zip

Unzip the paramiko package and then run the paramiko-1.7.2.win32.exe executable file found in the dist folder inside the paramikdo-1.7.2 folder

Pycrypto 2.0.1.win32-py2.4 (required by Paramiko)

A collection of cryptographic algorithms and protocols, implemented for use from Python.

http://www.amk.ca/python/code/crypto
(Main distribution site)

http://www.voidspace.org.uk/python/modules.shtml
(Windows binary distribution)

In order for Pycrypto and Paramiko to work, you will need a standard Unix key pair. You can use an existing key pair or you can run ssh-keygen on a Unix machine to produce a new key pair. Paramiko expects a private key using the standard the Unix key format used by opensssh. Note that some Windows ssh client applications use a different format, but allow you to import and export openssh format keys. If you have a key pair that you are already using for something else, you may use it for communication between iSiLK and SiLK.

NOTE: The Windows version of Paramiko may not work with all key types supported by openssh. In particular, version 1.7.2 does not support RSA keys (the openssh default), so you should be sure to specify the key type as DSA instead.

In either case, the public key should be appended to the authorized keys file on the Unix machine where the Silk Toolset is running, (usually in .ssh directory under home). The private key should reside on the windows machine in a known location to be used later in configuring iSiLK.

3. **Verify that you can run Python and load these modules.**
   Open a Command Prompt run python

```
C:\> cd c:\
C:\> python
```

From the python prompt, type the following, one after the other:

```
>>> import wx
>>> import matplotlib
>>> import numpy
>>> import paramiko
```

If all is well, these should all return without displaying an error message.

Step 2 – Install Python Source Code

After installing the prerequisites install the Python code for iSiLK. Simply create a directory for the source code and unpack it. For example, on Windows:

```
C:\> cd c:\
C:\> pkunzip isilk.zip
```

This will create a directory with the contents of the distribution.

Step 3 – Test your SSH connection

Before running iSiLK for the first time make sure that you can invoke SiLK tools using a standard command-line ssh client. This will allow you verify that SiLK is installed properly, that your PATH is properly set, and that the SSH setup is correct. For example, run:

```
C:\> sshclient.exe –i id_mykey me@host rwfilter –help
```

You can then verify rwfilter runs and that its output is the same as you see when you log with an ssh console application and run that command directly at the Unix command prompt.

Step 4 - Run iSiLK with Python

Launching iSiLK is simply a matter of running "isilk.py" with your python interpreter. For example, on a Windows host this may look something like:

```
C:\isilkdir> python isilk.py [options ]
```

See the iSiLK **User's Guide** for details on running iSiLK for the first time with your SiLK installation.

Building a Windows Binary Distribution

Introduction

The following steps are required to create a binary distribution for Windows. All of the following should be done on an installation of Windows XP.

Building the Windows Installer

1. **Install the *py2exe* tool.**
   This will allow you to create a directory that contains a standalone executable and the supporting files that will be installed in the "Program Files" directory.

   Py2Exe 0.6.6 (required to build .exe)

   http://sourceforge.net/project/showfiles.php?group_id=15583

2. **Run py2exe to generate the "dist" directory**

   ```
   C:\> cd isilk
   C:\isilk> python setup-windows.py py2exe
   ```

3. **Install the free version of Advanced Installer:**

   http://www.advancedinstaller.com/downloading.html

4. **Update the version number**

   If the version number has changed, choose "Product Details" from the "Project Settings" from the menu displayed to the left of the main window and update the "Product Version Field". It is recommended you respond "No" the dialog box that asks whether to create a new Product Code, since iSiLK is not designed to support side-by-side installs of multiple versions of the application.

5. **Generate the "isilk.msi" file using Advanced Installer:**
   The source distribution contains an Advanced Installer project file "isilk.aip". After installing the Installer, open it and choose File Open and select the isilk.aip file. Then select the "File and Folders" view in the "Project Definition" menu displayed to the left of the application window. This will show you all the files and sub-directories that will be copied to the installation directory when iSiLK is installed. You'll want to delete the existing contents of that directory before starting.

   Then, choose the "Application Folder" in the tree of folders displayed, and then add contents to that folder:

   - Select "Add Files" and add all the files in the `dist/` directory to the top-level "Application Folder" for the project.
   - For each folder in the `dist/` directory, use "Add Folder" and add this folder and its contents to the to the "Application Folder".

Finally, build the .msi file by selecting “Build” from the toolbar or the “Project” menu.

You should be able to run "isilk.msi" on any Windows XP or Vista system. The installer will install files in "Program Files", and create shortcuts for isilk.exe on your Start Menu.

NOTE: You will need to make sure the dll MSVCP71.dll is installed on your target system. This is generally copied to c:\Windows\System32 as part of the installation process of many Microsoft applications. It’s also distributed with most of the Microsoft developer tools. Before redistributing this file or any other file from Microsoft, be sure to consult the appropriate Microsoft license agreement.

Deployment Considerations

Where iSiLK Stores Files

Local Configuration Files

The first time you run iSiLK it will create the following configuration files:

```
C:\Documents and Settings\JDoe\Application Data\iSiLK\silk.conf
C:\Documents and Settings\JDoe\Application Data\iSiLK\library.dat
```

Local Data Files

Every active problem set corresponds to a directory on the user's local system in his documents directory. By default on Windows, problem sets are stored in the "isilk" sub-directory under "My Documents". This directly will typically look something like:

```
C:\Documents and Settings\JDoe\My Documents\isilk
```

iSiLK will create a subdirectory in this location for each problem set that has been saved. That problem set directory contains ascii versions of any of the files you've chosen to download, as well as a graphics file corresponding to every rendered graph in the problem set. It also contains an xml file, index.xml that describes the problem set and includes various iSiLK-specific meta-information. If you list a problem set directory on Windows it will look something like:

```
C:\Documents and Settings\JDoe\My Documents\isilk\jvt0.isilk>dir
 Volume in drive C has no label.
 Volume Serial Number is E089-36DF

 Directory of C:\Documents and Settings\JDoe\My Documents\isilk\jvt0.isilk

01/30/2008  06:30 PM    <DIR>          .
01/30/2008  06:30 PM    <DIR>          ..
01/30/2008  06:30 PM             9,449 Graph_-_Bytes-014p.png
01/30/2008  06:29 PM             9,665 index.xml
01/30/2008  06:28 PM           429,760 Untitled_Count_by_sport-56eu.asc
01/30/2008  06:29 PM        12,604,326 Untitled_Query-f39t.rwf.asc
01/30/2008  06:29 PM        12,604,326 Untitled_Refinement-srgj.rwf.asc
               5 File(s)     25,657,526 bytes
               2 Dir(s)  29,593,288,704 bytes free
```

Remote Data Files

The output directory that SiLK uses to store your remote results is configurable. For each local problem set directory iSiLK creates a corresponding remote problem set directory. The problem set directory includes a binary and ascii version of every output file generated during the course of analysis.

Note that it is user's responsibility to explicitly delete any remote data after it is no longer needed since iSiLK does not currently include features for deleting and archiving data.