



A Graphical Method for Running the ARL Secure Link Middleware

by Binh Q. Nguyen

ARL-TR-4353

January 2008

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-4353

January 2008

A Graphical Method for Running the ARL Secure Link Middleware

Binh Q. Nguyen

Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) January 2008		2. REPORT TYPE Final		3. DATES COVERED (From - To) Fiscal year 2007	
4. TITLE AND SUBTITLE A Graphical Method for Running the ARL Secure Link Middleware				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Binh Q. Nguyen				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRD-ARL-CI-CN 2800 Powder Mill Road Adelphi, MD 20783-1197				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-4353	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. National Archives and Records Administration 8601 Adelphi Road College Park, Maryland 20740-6001				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This report documents the results of the development of a graphical method for running the U.S. Army Research Laboratory (ARL) Secure Link middleware. It describes the internal implementation of the graphical user interface (GUI) and the methods for running the ARL Secure Link middleware. The report also includes the screen shots of the GUI and describes the functionality of the interfaces.					
15. SUBJECT TERMS Python, Tkinter, graphical user interface, GUI, Secure Link					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES 26	19a. NAME OF RESPONSIBLE PERSON Binh Q. Nguyen
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) 301-394-1781

Contents

List of Figures	iv
1. Introduction	1
2. Methods	1
3. Graphical User Interfaces	3
4. Results and Discussion	9
5. Conclusions	10
Appendix A. The Contents of the Configuration File	11
Appendix B. An Example of the Log File	13
Appendix C. The Appearance of the GUI in Microsoft Windows Environments	15
Acronyms	17
Distribution List	18

List of Figures

Figure 1. The initial appearance of the GUI of the ARL Secure Link middleware.....	3
Figure 2. Tkinter widgets.....	4
Figure 3. The pull-down menus.	5
Figure 4. Configuration parameters.	5
Figure 5. The Help menu and its contents.	6
Figure 6. The appearance of the GUI during operation.	7
Figure 7. The dialog showing missing files.	8

1. Introduction

To support the National Archives and Records Administration (NARA) in its effort to establish a secure distributed processing of electronic records archives (ERA), the U.S. Army Research Laboratory (ARL) developed an experimental information-assurance product, ARL Secure Link middleware¹, which is capable of providing data confidentiality between two networked computers. ARL has also developed a convenient method for running the middleware using a set of graphical user interfaces (GUIs).

This report documents the results of the development of the GUI, describes its features, and reveals its internal operations. The intended readers of this report include technical and managerial personnel as well as anyone who is interested in using the ARL Secure Link middleware or finding out how the GUI was built.

Section 2 describes the internal implementation of the GUI and the methods for running the ARL Secure Link middleware. Section 3 includes the screen shots of the GUI and describes the functionality of the interfaces. Section 4 discusses the development of the tool. Section 5 concludes the report.

2. Methods

The GUI developed for running the ARL Secure Link middleware was created using the **Python** programming language², its standard **Tkinter** module, and the **Popen** class of the **subprocess** module. The **Popen** class facilitates the interaction between the GUI and its execution environment by providing a means to control the operation of the middleware and its associated firewall. The paragraphs that follow describe the use of the **Popen** class in the GUI.

- Starting and stopping the ARL Secure Link middleware. The middleware was designed to run in the background as a daemon; hence it was named **sld** to stand for Secure Link daemon. A daemon is a computer program running in the background and requires no user interactions. Upon a successful execution of the **sld**, its running process identification (pid) is captured and saved for subsequent uses such as sending a signal to the running process:

pid = Popen(sld).pid

¹ Luu, Brian. *Functional Requirements Assessment of Secure Link*, ARL-MR-663; U.S. Army Research Laboratory, Adelphi, MD, March 2007.

² The Python Software Foundation, <http://www.Python.org> (accessed 15 Oct 07).

To stop a running **sld**, the GUI can send the **kill -s SIGKILL pid** or the **killall sld** command. The latter is the more preferable method for gracefully stopping all active sld processes because the **killall** command sends the SIGTERM signal (by default) that can be caught by the running processes. Upon receiving the SIGTERM signal, each **sld** process would clean up its work before terminating. Whereas, using the SIGKILL signal terminates **sld** processes abruptly, giving them no chance to close an opened file or a network communication socket, and thus leaving behind the system at an undetermined state.

- Enabling and disabling the firewall. The **iptables** command is used as the firewall with which the ARL Secure Link middleware operates. The command is often executed with specific instructions and options. To monitor the internal tables of the command (the state of the firewall), the GUI connects and redirects its outputs, including error messages, to an internal storage, **s**, by executing the following two statements:

Statement 1. **p = Popen(<iptables-command>, shell=True, stdout=PIPE, stderr=STDOUT,close_fds=True).stdout**

Statement 2. **s = p.read()**

Statement 1 executes a specific **iptables** command, and statement 2 retrieves the outputs of the executed command. The outputs are immediately displayed in the GUI.

3. Graphical User Interfaces

The GUI of the ARL Secure Link middleware was built using the standard widgets readily available in the **Tkinter** library, which is a **Python** extension to the **tcl** programming language and the **tk** GUI toolkit.³ When the GUI is executed, it appears on the computer screen as shown in figure 1.

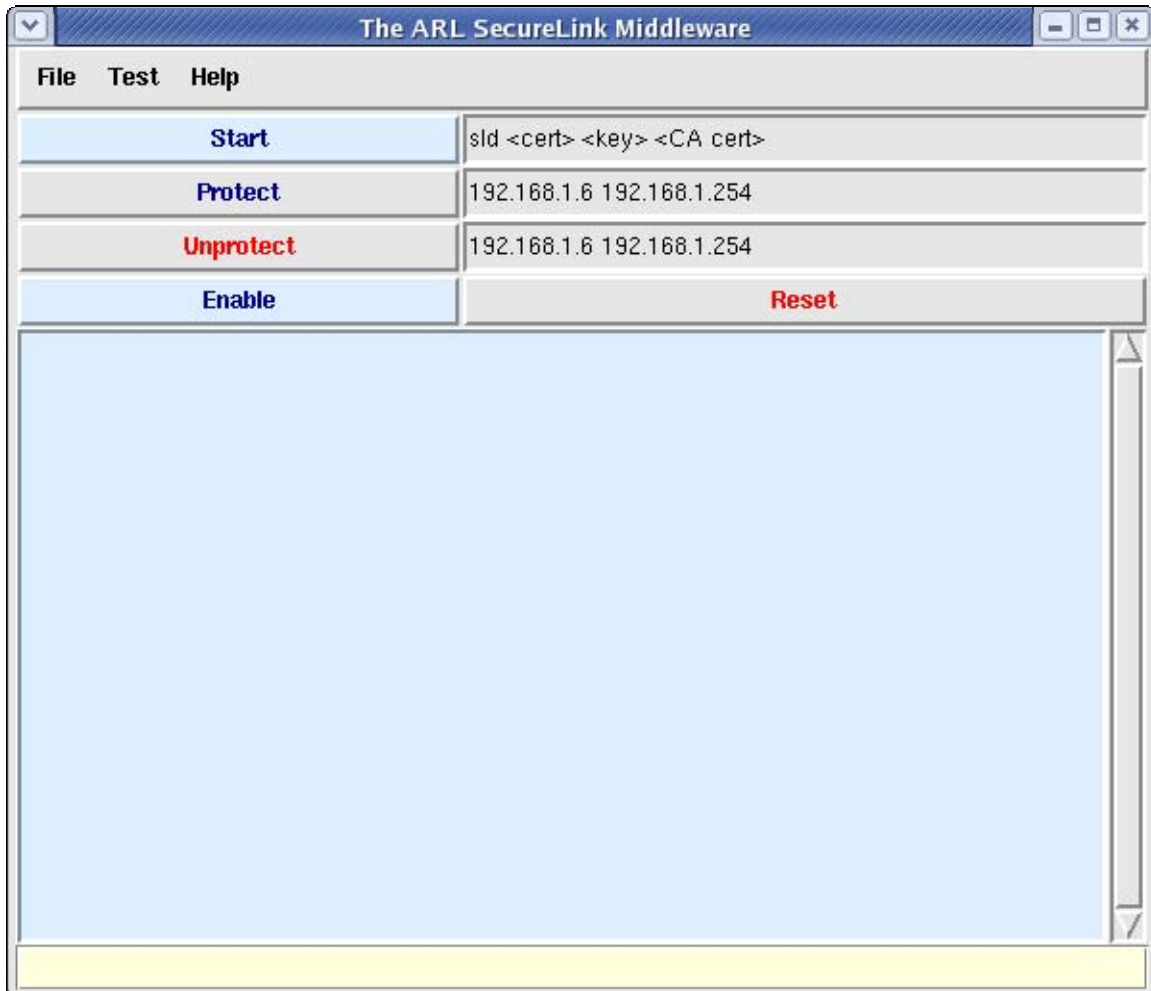


Figure 1. The initial appearance of the GUI of the ARL Secure Link middleware.

³ Tcl Developer Xchange, <http://www.tcl.tk> (accessed 15 Oct 2007).

Important widgets that were used to build the GUI include the **Frame**, **Menubar**, **Button**, **Entry**, **Text**, **Scrollbar**, and **Label** widgets. The **Frame** widget contains other widgets, including other **Frame** widgets. Figure 2 shows the **Tkinter** widgets that were used to build the GUI.

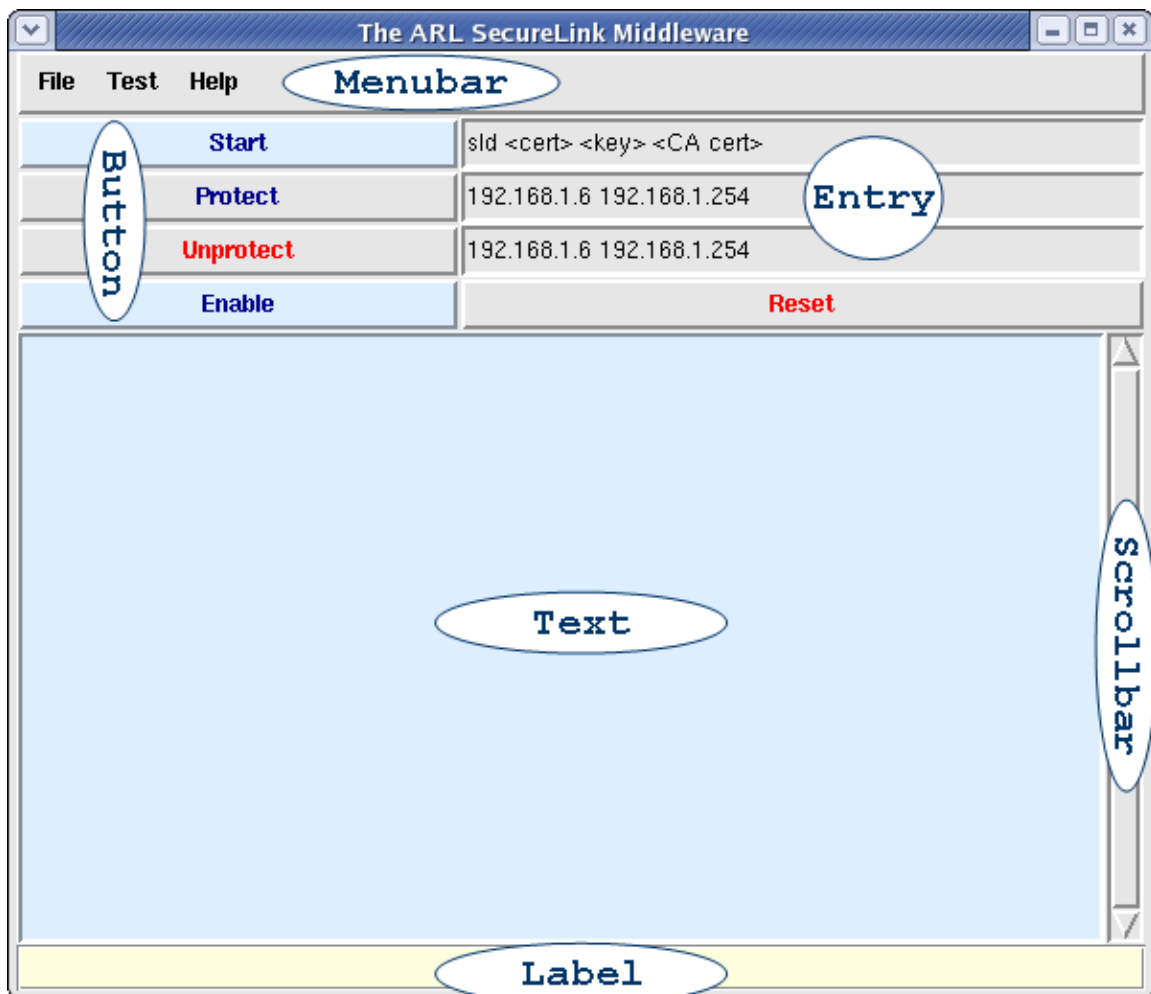


Figure 2. Tkinter widgets.

On the top of the GUI is the **Menubar** with three pull-down menus containing features infrequently used during the operation of the ARL Secure Link middleware. However, the user has an option to “tear off” the menus to make them permanently displayed until they are manually destroyed. The torn-off menus are depicted in figure 3.

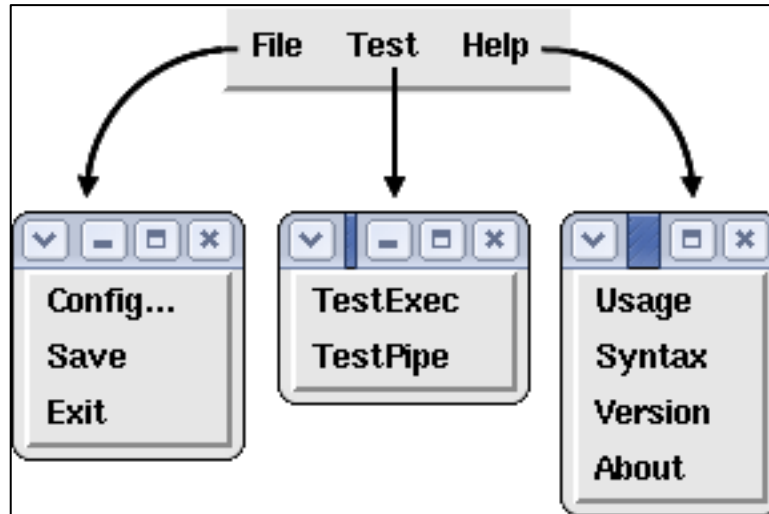


Figure 3. The pull-down menus.

The three pull-down menus include two menus for the users (**File** and **Help**) and one menu for the developer (**Test**). The File menu has three menu items: **Config...**, **Save**, and **Exit**. Selecting the **Exit** menu item shuts down the operation of the ARL Secure Link middleware, resets the state of the firewall, closes the log file, and destroys the GUI. Selecting the **Save** menu item stores the currently configured options to a file whose name is a configuration parameter. Selecting the **Config...** menu item displays a table of configuration parameters and their current values as shown in figure 4. A new value can be entered in the **Entry** box of the parameter. Pressing the **OK** button confirms the change. Pressing the **Cancel** button closes the window and abandons the change.

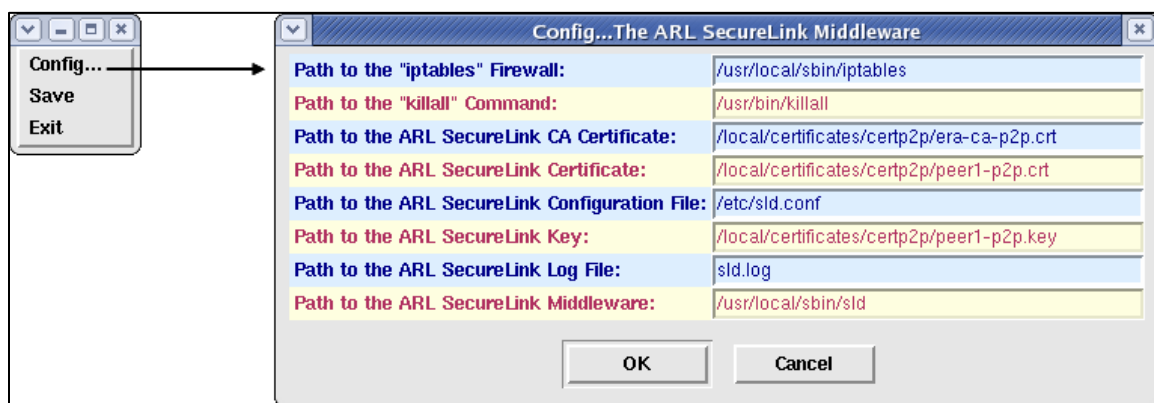


Figure 4. Configuration parameters.

The **Test** pull-down menu is used during the development of the GUI to experiment with new features. Selecting any item of the **Test** menu does not affect the operation of the ARL Secure Link middleware.

The **Help** pull-down menu has four different informational types: **Usage**, **Syntax**, **Version**, and **About** as shown in figure 5. Selecting the **Usage** menu item gives a short set of instructions on how to run the ARL Secure Link middleware using the GUI. Selecting the **Syntax** menu item displays the ordered list of required parameters for running the middleware. Selecting the **Version** menu item shows the version and the date of the GUI. Selecting the **About** menu item shows a blurb about the tool.

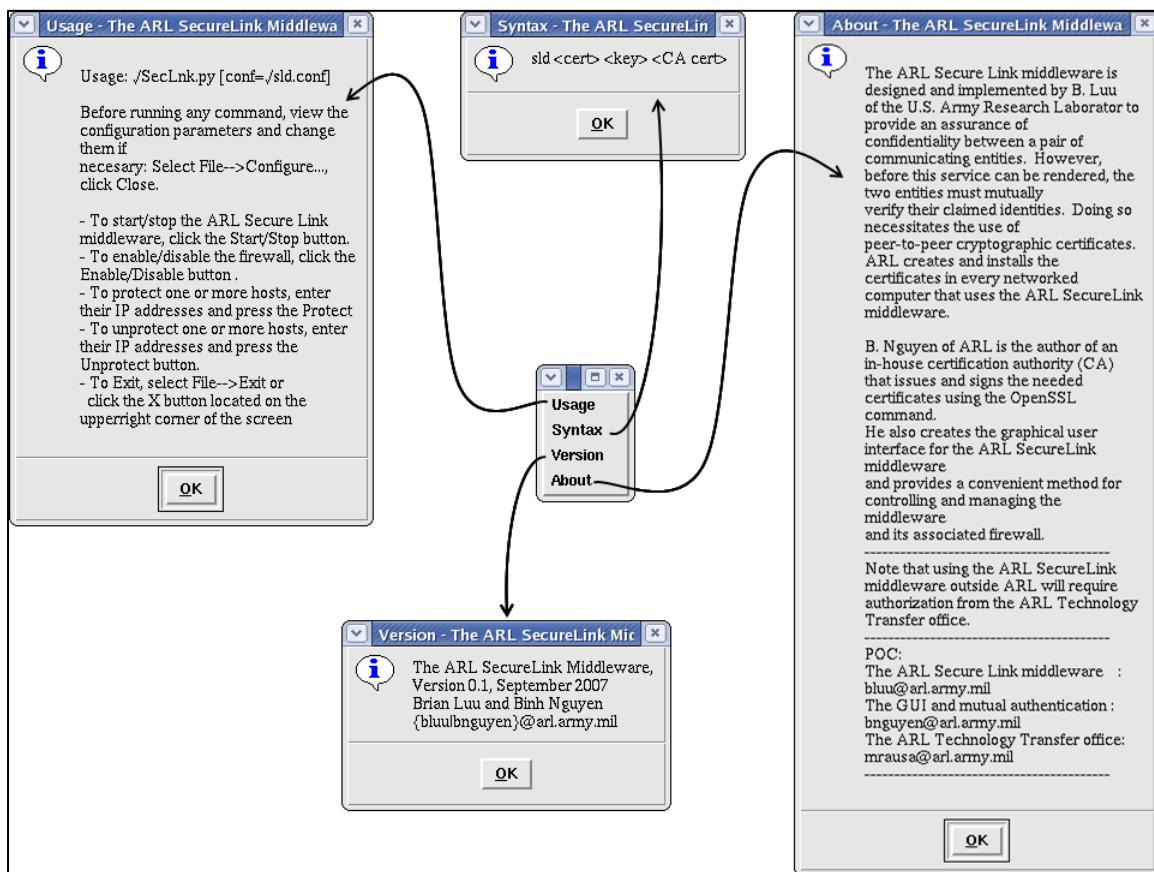


Figure 5. The Help menu and its contents.

Below the pull-down menus, the GUI is divided into three sections. The first section provides the user a way to interact with the ARL Secure Link middleware. It has five action buttons and their related **Entry** boxes. The buttons are labeled **Start/Stop**, **Protect**, **Unprotect**, **Enable/Disable**, and **Reset**. Pressing the **Reset** button terminates the running ARL Secure Link middleware, clears the internal tables of the firewall, and erases the display area of the GUI. Pressing the **Enable** button initializes the state of the firewall and changes its label from **Enable** to **Disable**. Pressing the **Disable** button resets the firewall and changes its label back to **Enable**.

The **Unprotect** button requires a list of one or more internet protocol (IP) addresses to be entered and displayed in its corresponding **Entry** box, located on the right side of the button. Pressing the **Unprotect** button or pressing the **Enter** key while the mouse cursor is in the **Entry** box instructs the ARL Secure Link middleware to stop providing confidentiality services to the host(s) having their addresses displayed in the **Entry** box. Similarly, the **Protect** button is used to instruct the middleware to provide confidentiality services to the host(s) having their addresses displayed in the corresponding **Entry** box. The lists of unprotected and protected hosts need not be identical. In order for the two buttons to function properly, the **sld** daemon must be running and the firewall must be enabled.

The **Start** button is used to run the **sld** daemon with its predetermined configuration parameters or running it with a different set of cryptographic certificates shown in its corresponding **Entry** box. Once the **sld** daemon has been executed, the button changes its label from **Start** to **Stop** and then enables the firewall as if the **Enable** button were pressed by the user as shown in figure 6.

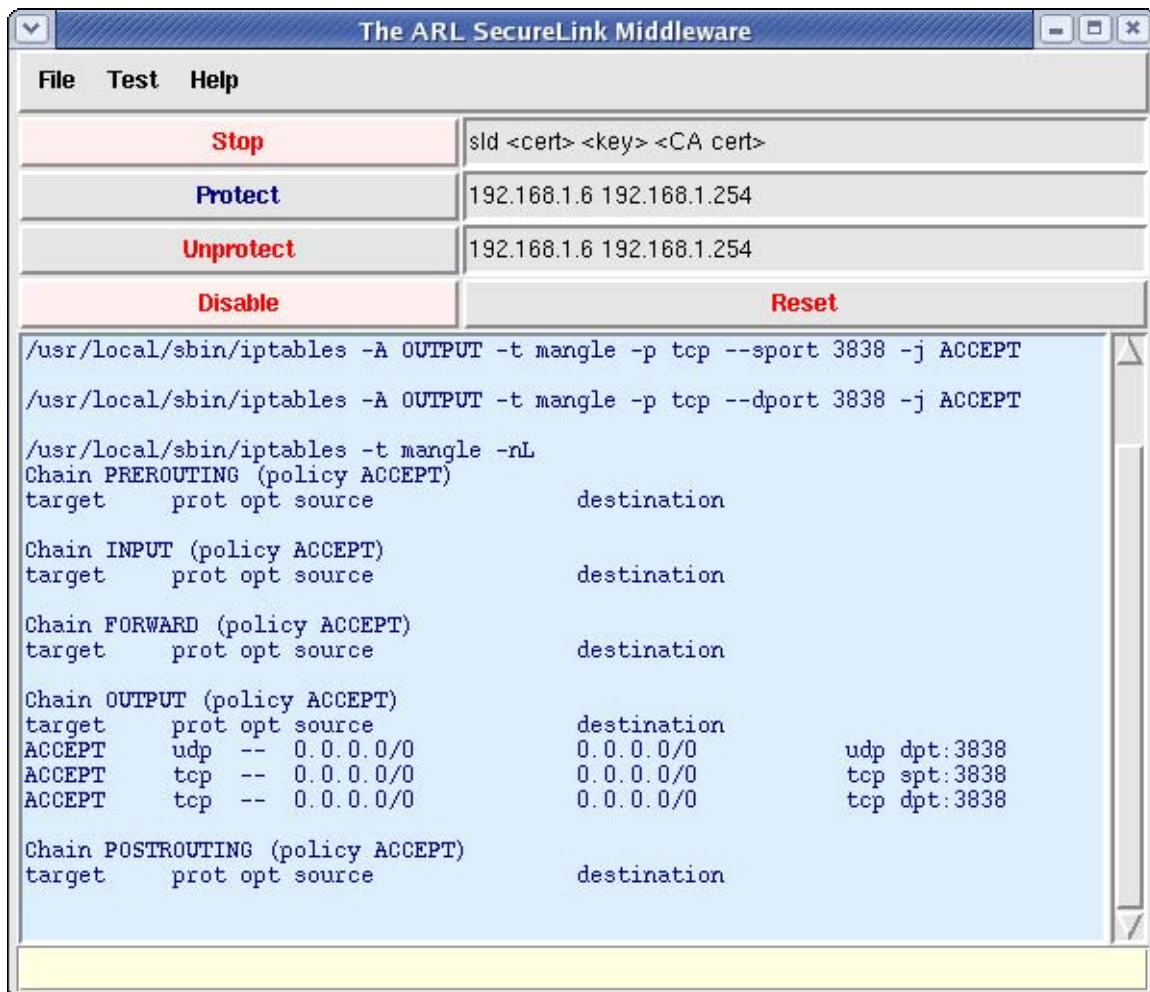


Figure 6. The appearance of the GUI during operation.

The second section of the GUI is the largest area that uses the **Text** and **Scrollbar** widgets to display the executed commands and their outputs, which provides the user a way to visually monitor the status of the firewall immediately after every execution of a command. In addition to displaying the history and outputs of the executed commands, the GUI also writes to a log file the date and time of an executed command and its outputs. Pressing the **Reset** button clears the display area, but does not clear the log file. The contents of the log file are subsequently used for post-analysis of user-initiated events and the status of the firewall to support the development and improvement of the ARL Secure Link middleware. If a log file is unwanted, then the name **/dev/null** must be set as its name. The GUI will continue sending logged messages to **/dev/null**, but the user will not be able to see and keep them.

The third section is a thin strip located at the bottom of the GUI used to provide user-feedback messages. It is often known as the status bar.

The operation of the GUI consists of two sequential steps: (1) initialization, and (2) waiting for a user-initiated event. During the initialization phrase, the GUI checks for the existence of the given configuration file, which has specifications of paths to the vital files that ARL Secure Link middleware needs to function. (Appendix A provides an example of the configuration file.) If the given configuration file exists and is readable, then the GUI reads in its contents to parse and extract the configured parameter values. If the configuration file does not exist, the GUI uses its default values that can be subsequently changed by the user. The GUI then verifies the existence of the configured values by checking the specified paths to the files and creating a list of invalid paths. At the end of the verification, if the list is non-empty, the GUI opens a dialog window showing the invalid paths and asking the user to decide whether the GUI should exit or continue running. Figure 7 depicts the dialog showing the name of the configuration file that does not exist during the initialization phase. Pressing the **No** button terminates the GUI. Pressing the **Yes** button lets the GUI continue running (mainly for exploration and learning purposes).

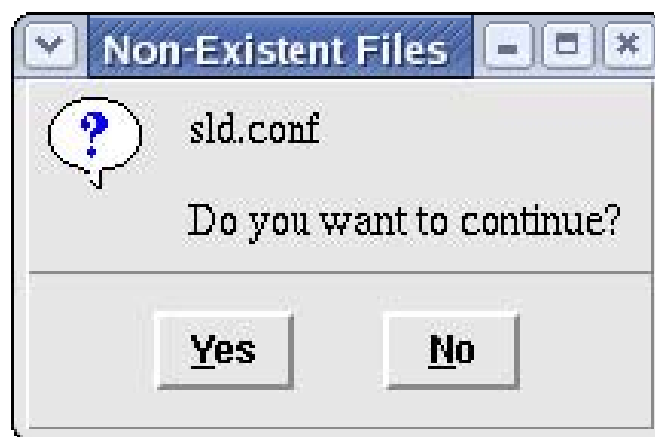


Figure 7. The dialog showing missing files.

Throughout the session, the GUI simply waits for the user to select one of its built-in features. Each selected feature creates an event that drives the GUI into action. The following events are suggested for running the GUI-equipped ARL Secure Link middleware.

- Verify that all values of the configuration parameters are correct by selecting the pull-down menu item **File:Config...**
 - Start the ARL Secure Link middleware and enable its associated firewall by pressing the **Start** button.
 - Provide confidentiality services to a set of hosts by entering their IP addresses into the **Entry** box of the **Protect** button and pressing the **Enter** key or **Protect** button.
 - Deprive one or more hosts of confidentiality services by entering their IP addresses into the **Entry** box of the **Unprotect** button and pressing the **Enter** key or **Unprotect** button.
 - Terminate the ARL Secure Link middleware without leaving the GUI by pressing the **Stop** or **Reset** button.
-

4. Results and Discussion

Using the GUI to run the ARL Secure Link middleware was simple. Learning how to use it could be completed in about 5 minutes. The GUI simplifies the learning, training, and operating purposes. All the configuration parameters are readily available to its users to view and to change, if necessary. All the locations of the vital files needed for a successful session were clearly specified and presented to the user. The whole process of creating a secure communications among the participating networked computers consists of three major steps: (1) running the middleware, (2) deciding the hosts that will receive the confidentiality services and the hosts that will not, and (3) terminating the middleware and reset the system.

The GUI was successfully built and run on Linux environments where the ARL Secure Link middleware and the **iptables** packet-filtering system operate. Besides providing convenient methods for running the two system tools, the GUI offers a way to edit and manage important configuration parameters. It also automatically counts and records user-initiated commands, captures the outputs of the executed commands, and creates a log file storing historical events in a session (Appendix B) for subsequent analysis, debugging, training, or reporting purposes.

The use of the **Python** programming language enables the GUI to run in any operating system that hosts a current **Python** interpreter. In fact, ARL developed the GUI in a Microsoft Windows environment, but tested it in a Linux environment where the ARL Secure Link operates. Appendix C shows the appearance of the GUI in a Microsoft Windows environment.

The use of the **Python** programming language and its **Tkinter** modules saves resources that would have been spent to develop a separate GUI for each operating system.

Had a version of the ARL Secure Link middleware been developed to run in a Microsoft Windows environment, then the GUI would have been ready to interact with the middleware, requiring no changes to the core structural components of the GUI and a few changes to the system interfaces to accommodate the idiosyncrasy between the two different operating systems.

5. Conclusions

The GUI and the methods for running the ARL Secure Link middleware represent a convenient way for dealing with system and application software. The use of the **Python** programming language really expedites the development process, thereby increasing programming productivities and shortening development time. Its standard built-in GUI modules were easy to integrate into a complete system, thereby voiding the need of a GUI development tool. This source code of the GUI can serve as a template for designing the GUI for other applications.

Appendix A. The Contents of the Configuration File

```
#This file is automatically generated by the
#GUI of the ARL Secure Link Middleware
#
#=====
#@Path to the ARL SecureLink Configuration File:=sld.conf
#@Path to the ARL SecureLink Log File:=sld.log
#@Path to the ARL SecureLink Key:=/local/pki/certp2p/peer1-p2p.key
#@Path to the "killall" Command:=/usr/bin/killall
#@Path to the "iptables" Firewall:=/usr/local/sbin/iptables
#@Path to the ARL SecureLink Middleware:=/usr/local/sbin/sld
#@Path to the ARL SecureLink CA Certificate:=/local/pki/certp2p/era-ca-p2p.crt
#@Path to the ARL SecureLink Certificate:=/local/pki/certp2p/peer1-p2p.crt
#@Protected IP addresses:=192.168.1.6 192.168.1.254
#
#===== requested variables:
#
syskey="/local/pki/certp2p/peer1-p2p.key"
syscacert="/local/pki/certp2p/era-ca-p2p.crt"
syscert="/local/pki/certp2p/peer1-p2p.crt"
protect="192.168.1.6 192.168.1.254"
#Mon Oct 15 10:52:30 2007
```

INTENTIONALLY LEFT BLANK.

Appendix B. An Example of the Log File

Wed Oct 17 11:09:15 2007: 1 /usr/local/sbin/sld /local/pki/certp2p/peer1-p2p.crt /local/pki/certp2p/peer1-p2p.key
/local/pki/certp2p/era-ca-p2p.crt

Wed Oct 17 11:09:15 2007: 2 /usr/local/sbin/iptables -A OUTPUT -t mangle -p udp --dport 3838 -j ACCEPT

Wed Oct 17 11:09:15 2007: 3 /usr/local/sbin/iptables -A OUTPUT -t mangle -p tcp --sport 3838 -j ACCEPT

Wed Oct 17 11:09:15 2007: 4 /usr/local/sbin/iptables -A OUTPUT -t mangle -p tcp --dport 3838 -j ACCEPT

Wed Oct 17 11:09:15 2007: 5 /usr/local/sbin/iptables -t mangle -nL

Chain PREROUTING (policy ACCEPT)

target	prot	opt	source	destination
--------	------	-----	--------	-------------

Chain INPUT (policy ACCEPT)

target	prot	opt	source	destination
--------	------	-----	--------	-------------

Chain FORWARD (policy ACCEPT)

target	prot	opt	source	destination
--------	------	-----	--------	-------------

Chain OUTPUT (policy ACCEPT)

target	prot	opt	source	destination
ACCEPT	udp	--	0.0.0.0/0	0.0.0.0/0 udp dpt:3838
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0 tcp spt:3838
ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0 tcp dpt:3838

Chain POSTROUTING (policy ACCEPT)

target	prot	opt	source	destination
ROUTE	0	--	0.0.0.0/0	192.168.1.6 ROUTE oif:sld
ROUTE	0	--	0.0.0.0/0	192.168.1.254 ROUTE oif:sld

...

...

Chain POSTROUTING (policy ACCEPT)

target	prot	opt	source	destination
--------	------	-----	--------	-------------

Wed Oct 17 11:09:51 2007: 21 /usr/bin/killall sld

Wed Oct 17 11:09:51 2007: 22 /usr/local/sbin/iptables -D OUTPUT -t mangle -p udp --dport 3838 -j ACCEPT

Wed Oct 17 11:09:51 2007: 23 /usr/local/sbin/iptables -D OUTPUT -t mangle -p tcp --sport 3838 -j ACCEPT

Wed Oct 17 11:09:51 2007: 24 /usr/local/sbin/iptables -D OUTPUT -t mangle -p tcp --dport 3838 -j ACCEPT

Wed Oct 17 11:09:51 2007: 25 /usr/local/sbin/iptables -t mangle -nL

Chain PREROUTING (policy ACCEPT)

target prot opt source destination

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot opt source destination

Chain OUTPUT (policy ACCEPT)

target prot opt source destination

Chain POSTROUTING (policy ACCEPT)

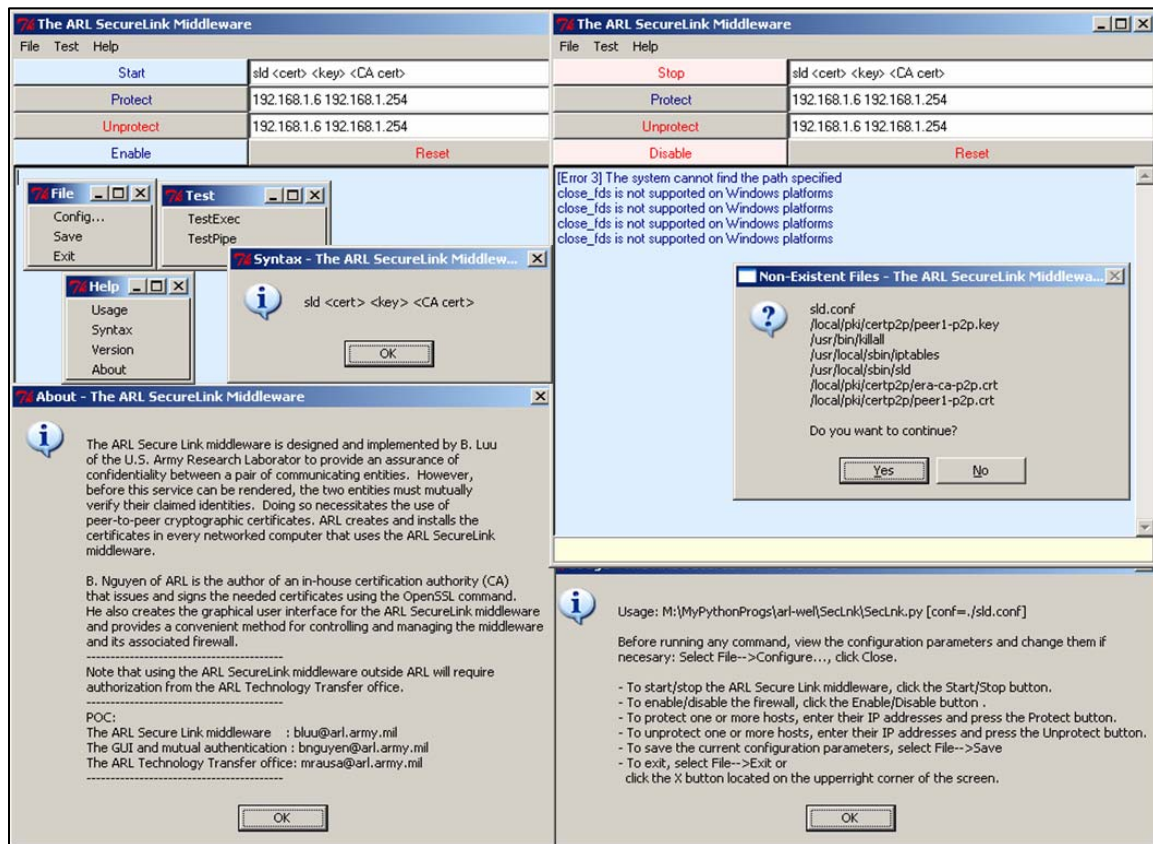
target prot opt source destination

...

...

Wed Oct 17 11:10:03 2007: 40 /usr/bin/killall sld

Appendix C. The Appearance of the GUI in Microsoft Windows Environments



INTENTIONALLY LEFT BLANK.

Acronyms

ARL	U.S. Army Research Laboratory
ERA	electronic records archives
GUIs	graphical user interfaces
IP	internet protocol
NARA	National Archives and Records Administration
pid	process identification

Distribution List

No. of Copies	<u>Organization</u>	No. of Copies	<u>Organization</u>
1 elec	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP (ELECTRONIC COPY) 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218	1 HC	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1 HC	DARPA ATTN IXO S WELBY 3701 N FAIRFAX DR ARLINGTON VA 22203-1714	1 HC	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402
1 HC	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080	1 HC	DIRECTOR US ARMY RSRCH LAB ATTN AMSRD ARL RO EV W D BACH PO BOX 12211 RESEARCH TRIANGLE PARK NC 27709
1 HC	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV AND ENGRG CTR ARMAMENT ENGRG AND TECHN LGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 APG MD 21005-5001	12 HCs	US ARMY RSRCH LAB ATTN IMNE ALC IMS MAIL & RECORDS MGMT ATTN AMSRD ARL D J M MILLER ATTN AMSRD ARL CI OK TL TECHL LIB (1 COPY) ATTN AMSRD ARL CI OK T TECHL PUB (2 COPIES) ATTN AMSRD ARL CI G RACINE (2 COPIES) ATTN AMSRD ARL CI CN B RIVERA (1 COPY) ATTN AMSRD ARL CI CN R GOPAUL (1 COPY) ATTN AMSRD ARL CI CN B LUU (2 COPIES) ATTN AMSRD ARL CI CN B NGUYEN (2 COPIES) 2800 POWDER MILL ROAD ADELPHI MD 20783-1197
1 HC	US ARMY TRADOC BATTLE LAB INTEGRATION & TECHL DIRCTRT ATTN ATCD B 10 WHISTLER LANE FT MONROE VA 23651-5850		
1 HC	PM TIMS, PROFILER (MMS-P) AN/TMQ 52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703		
1 HC	SMC/GPA 2420 VELA WAY STE 1866 EL SEGUNDO CA 90245-4659	1 HC	US ARMY RESEARCH LAB AMSRD CI OK TP TECHL LIB ATTN T LANDFRIED APG MD 21005
1 HC	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD F JENIA FT HUACHUCA AZ 85613-5300		

No. of <u>Copies</u>	<u>Organization</u>
5 HCs	ROBERT CHADDUCK
1 Elec	US NATIONAL ARCHIVES & RECORDS ADMIN 8601 ADELPHI RD COLLEGE PAR MARYLAND 20740-6001

Total: 30 (2 Electronic, 28 HCs)

INTENTIONALLY LEFT BLANK.