Building GUI Software for Managing CISCO Network Appliances

A Manuscript
Submitted to
the Department of Computer Science
and the Faculty of
University of Wisconsin-La Crosse
La Crosse, Wisconsin

by

Quoc M Le

in Partial fulfillment of the
Requirements for the degree of

Master of Software Engineering

June 2008

Building GUI Software for Managing CISCO Network Appliances

By Quoc M Le

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

_____	_____
Dr. David Riley Examination Committee Chairperson	Date
_____	_____
Dr. Kasi Periyasamy Examination Committee Member	Date
_____	_____
Dr. Kenny Hunt Examination Committee Member	Date

ABSTRACT

LE, QUOC, M., "Software For Managing Cisco Network Appliances", Master of Software Engineering, Aug 2008, Advisors: Dr. David Riley, Dr. Kasi Periyasamy.

The goal of this project is to build GUI software for managing Cisco network appliances. This system includes a built-in ACL parser to parse ACLs and identify conflicts and redundancies in the configuration. The system also assists with convenient access list edits and provides for backup, restore, and versioning of device configuration. This work provides part of a tool for network administration and was designed and tested with the close supervision of a professional network administrator.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my project advisors Dr. David Riley and Dr. Kasi Periyasamy for their valuable guidance. I like to thank the project sponsor, Mr. Milo Velimirovic who initiated this project and provided the support for this project. I would also like to express my thanks to the Computer Science Department and the University of Wisconsin-La Crosse for providing the computing environment for my project. Finally, I wish to thank my parents, my sister and my friends for their patience and encouragement during the tenure of this project.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

GLOSSARY

GUI

Graphical User Interface (GUI) is the presentation layer for user to interact with application

CLI

Command Line Interface (CLI) is a mechanism for interacting with a computer operating system or system software by typing commands to perform specific tasks. This contrasts with the use of a mouse pointer with a GUI to click on options, or menus on a text user interface (TUI) to select options.

UML

Unified Modeling Language™ (UML) is an industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems standardized by the Object Management Group. UML simplifies the complex process of software design by using “blueprints” for software construction.

Use Case

A use case is a technique for capturing functional requirements of systems and systems-of-systems. Each use case provides one or more scenarios that convey how the system should interact with the users called actors to achieve a specific business goal or function.

IEEE

The Institute of Electrical and Electronics Engineers (IEEE) is an international organization whose constitution describes their purpose as "scientific and educational, directed toward the advancement of the theory and practice of several engineering fields including computer science."

Access list

An access list (ACL) is a list of policies that categorize packets. One of the most common and easiest to understand uses of access lists is filtering unwanted packets when implementing security policies. Each individual policy is known as an access list entry (ACE).

Software Prototyping

Prototypes provide software developers with a "working model" for demonstration or use by customers, quality-assurance, business analysts, and managers to confirm or make changes to requirements, help define interfaces, develop collaborating components, and to provide proof of incremental achievement of scheduled contractual agreements. Software Prototyping serves any and all of these purposes in practice.

Telnet

Telnet is a protocol that allows a user on a remote client machine, called the telnet client, to access the resources of another machine, the telnet server.

IP

IP is a network layer protocol in the Internet protocol suite and is encapsulated in a data link layer protocol (e.g., Ethernet). As a lower layer protocol, IP provides the service of communicable unique global addressing amongst computers.

IP Address

An IP address (or internet protocol address) is a unique address that certain electronic devices use in order to identify and communicate with each other on a computer network utilizing the Internet Protocol standard

IOS

CISCO IOS (originally Internetwork Operating System) is the software used on the vast majority of Cisco Systems routers and all current Cisco network switches. IOS is a package of routing, switching, internetworking and telecommunications functions tightly integrated with a multitasking operating system.

1. BACKGROUND

Today, networks are an essential part of business, education, government and home communications. Cisco is a worldwide leader in networking equipment. Cisco products include routers, switches, firewalls, VPN[1], and IDS[2] ... The Cisco IOS[3] is the kernel of routers and most other network appliances. IOS utilizes a powerful command line interface (CLI) for configuring Cisco routers and other similar devices. While CLI provides users with a great deal of control, it does not give them any capacity to verify that the rules they have formed are truly the rules they desire. CLI includes no built-in checks for semantic flaws, only minimal syntax checking.

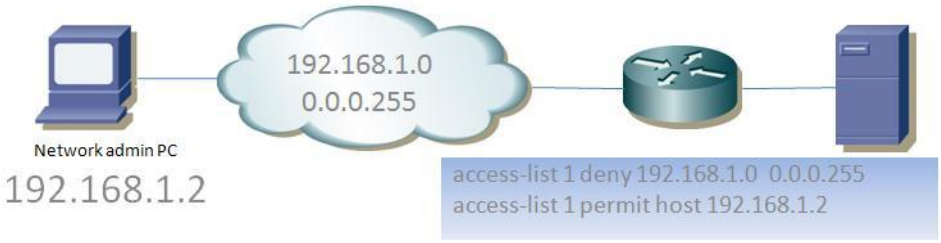
In general, network appliances, including those configured using CLI, base their security rules on an ordered sequence of access control lists (ACLs). An ACL is essentially a list of conditions that dictate packet access and blocking. With ACLs, a manager can implement security policies and protect sensitive devices from unauthorized access. When a request is received by a router, the ACLs are evaluated until an applicable rule is found. Once such a rule instance is discovered the search ends and access is either granted or denied, according to the rule. Due to the complexity of ACLs and the weakness of the command line interface that controls them, network administrators often create ACLs that include contradictory or redundant rules.

[1] VPN: Virtual Private Network
[2] IDS: Intrusion Detection System
[3] IOS: Internetwork Operating System

Example:

Policy #1: block any traffic from the 192.168.1.0 network, regardless of the packet's destination

Policy #2: allow traffic from 192.168.1.2 (network admin PC)



The above implementation of this policy fails to allow the administrator to connect to the server because the connection will be dropped as specified in the first policy or ACE. Reversing the order of the two ACEs corrects this problem. The mistake is not detected by the CLI (see Figure 1) but can easily be spotted by a network administrator. Unfortunately, it is difficult for humans to identify such issues when the access lists become more lengthy and complex, which is normal for real-world router configuration. Adding to the potential for configuration errors, there are many ways to specify the IP address of a host, including ranges, IP masks and wildcards; it can be very difficult for network administrator to verify access lists manually.

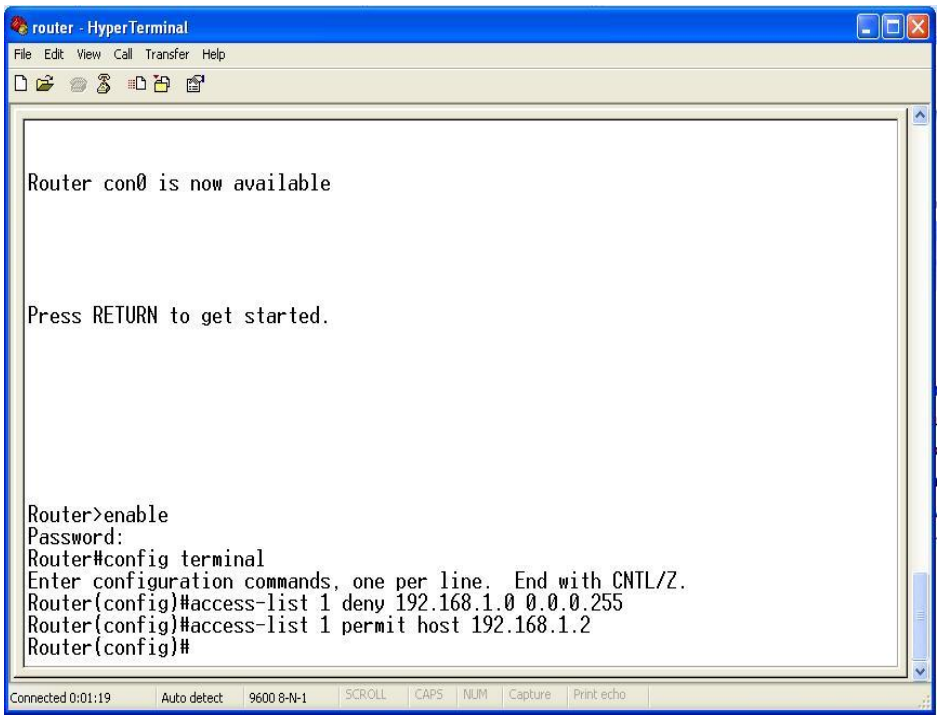


Figure 1: Configuration with conflict was accepted

This project is the development of software to analyze ACL semantics. The application parses each access list entry, comparing it with the rest of the access list. The algorithm detects any redundancies or conflicts in the access list.

Another issue with current tools for network appliance configuration is version control support. The management of configuration files can be difficult, time-consuming, and error-prone because the files tend to be large and complex, especially when they contain large numbers of ACLs. A user might update ACLs by command line interface with hundred of lines of commands. If the user makes a mistake, (conflicts or redundancies…) it may be necessary to manually undo all the changes that were made.

The software developed in this project stores the configuration details along with version details and user comments.

2. OVERVIEW OF CISCO IOS

2.1 IP address quick review

An IP, Version 4, address consists of 32 bits of information. These bits are divided into four octets. Each octet containing 8 bits (1 byte).

Decimal octet: 192.168.20.54

Dotted-binary: 11000000.10101000.00010100.00110110

2.2 Range of IP addresses quick review

A range of IP addresses can be specified by an IP with wildcard mask. Wildcards are used with access lists to specify an individual host, a network, or a certain range of a network or networks. Wildcard masks look like subnet masks, but they are not. The wildcard mask is the complement of the subnet mask, so that we can use below simple formula to computing the correct wildcard mask from any subnet mask:

Wildcard = 255 – subnet

For example, the wildcard mask that corresponds to a subnet mask of 255.255.255.0 is 0.0.0.255

To specify a host, the address looks like this: 172.16.30.5 0.0.0.0. The four zeros represent each octet of the address. Whenever a zero is present, it means that octet in the address must match exactly. To specify that an octet can be any value, the value of 255 is used. For example a 255.255.255.0 subnet is specified with a wildcard of 0.0.0.255. This tells the router to match up the first three octets exactly, but the fourth octet can be any value.

2.3 Access-list command quick review

An access list is a series of ACEs that the network device uses to control traffic flow into or out of an interface.

Each ACE must contains at least three important parts:

- A number that identifies the list (ACL) to which it belongs
- A “deny“ or “permit” instruction
- A packet identifier (an IP address or range of IP addresses)

Example:

access-list 1 deny 192.168.1.0 0.0.0.255

access-list 1 deny 100.10.2.0 0.0.0.255

access-list 1 permit any

There are a few important rules that a packet follows when it’s being compared with an access list:

- It’s always compared with each line (ACE) of the access list in sequential order—i.e., the device’s algorithm always start with the first line of the access list, then line 2, then line 3, and so on.

- Comparison with lines of the access list proceeds only until a match is made. Once the packet matches the condition on a line of the access list, the packet is acted upon and no further comparisons take place.

- There is an implicit “deny” at the end of each access list—this means that if a packet does not match the condition on any of the lines in the access list, the packet will be discarded.

2.4 Types of access lists

CLI includes two main types of access lists: standard access lists and extended access lists.

2.4.1 Standard access list

These use only the source IP address in an IP packet as the condition test. All decisions are made based on the source IP address. This means that standard access lists basically permit or deny an entire suite of protocols. They don't distinguish between any of the many types of IP traffic such as web, telnet, UDP, and so on.

Syntax: **access-list *number action source***

The parameters are:

number: A number between 1 and 99, identifying the list for future reference

action: The keyword "permit" or "deny", indicating whether to allow or block the packet

source: The packet's source address

Example:

access-list 10 deny host 172.16.30.2

This ACE does not allow any traffic from host 172.16.30.2.

2.4.2 Extend access list

Extended access lists can evaluate many of the other fields in the OSI layer 3 and OSI layer 4 headers of an IP packet. They can evaluate source and destination IP addresses, the protocol field in the network layer header, and the port number at the transport layer header. This gives extended access lists the ability to make much more granular decisions when controlling traffic.

Syntax: **access-list *number action protocol source s-port destination d-port* [*optional-args*]**

The action and source address are the same as for standard access lists. The other fields are:

number: A number identifying the list. For extended access lists, this number must be between 100 and 199.

protocol: An indication of the protocol to which the rule applies. This must be either ip, tcp, udp, or icmp.

s-port: For TCP or UDP packets, the packet's source port. There are a number of ways to specify ports. This field is optional. If the protocol is IP or ICMP, this field is omitted.

destination: The packet's destination address, specified the same way as the source address. That is, you can have an IP address followed by a wildcard mask, the keyword host followed by the IP address of a specific host, or the keyword any.

d-port: For TCP or UDP packets, the packet's destination port. There are a number of ways to specify ports. This field is optional. If the protocol is IP or ICMP, this field is omitted.

optional-args: An optional keyword that is applicable only if the protocol is TCP. For example, the keyword established is optional.

Example:

access-list 110 deny tcp any host 172.16.10.2 eq 23

The above ACE blocks all telnet (TCP, port 23) traffic to host 172.16.10.2

3. ANALYSIS AND DESIGN

3.1 Analysis

Software application development, commonly known as the Software Development Life Cycle, encompasses all activities (phases) to develop an application system until it is put into production. A software product goes through the series of phases: (i) requirements phase, (ii) specification phase, (iii) planning phase, (iv) design phase, (v) implementation phase, (vi) integration phase, (vii) maintenance phase. There are a wide variety of software life cycle models reported in the literature. Examples for life cycle models include waterfall, iterative, rapid, spiral, RAD, and many more. There is no single life cycle model that is best suited for every project. Different software life cycles may complement each other on the same project during different phases of development of a product. Successful use of a particular life cycle model does not ensure success or a best fit for a new project. Choosing the most appropriate model for a particular software project is challenging. Many factors such as team size, project size, available resources, deadlines or milestones, team skills and experience, business policies, and application domain may influence the choice of a life cycle model.

The GUI software for managing Cisco network appliances was developed as a standalone application. The first phase of development involved requirements gathering. In the second phase requirements were analyzed and a detailed specification was written. In the third phase, the product was implemented using the incremental prototype model. In the fourth phase the implemented

application was tested thoroughly by the developer and the project sponsor. Bugs were identified and fixed in this last phase.

During the first phase, the requirement phase, the developer worked with the project sponsor to determine precisely what he need. The project sponsor served as the application domain resource for answering various questions during the requirements gathering phase. After several weekly meetings with the sponsor during summer 2007 about project, the developer was able to extract initial requirements. During this summer, some prototypes were constructed and given to the project sponsor. These prototypes were used to understand how the program works and enable the project sponsor and developer to agree as quickly as possible on what the behavior of the product. The prototype was created with a simple GUI designed to be altered and to demonstrate a potential GUI to the project sponsor. In addition some new function was added at this time. Once the demo of the prototype was completed and the sponsor gave feedback, the requirements for the entire product were developed using a more formal SRS document. The SRS is a valuable communication tool that provided input into other phases, such as design and testing. The benefits of developing a prototype early in the software process are the following: (1) misunderstanding between software developers and users may be identified as the system functions are demonstrated; (2) missing functions may be detected; (3) confusing functions may be identified and refined; (4) the developer may find incomplete and/or inconsistent requirements as the prototype is developed.

From the initial set of requirements, the first prototype was made. This helped in determining whether the stated requirements were unclear, incomplete, ambiguous, or contradictory; such deficiencies were resolved. The second prototype was developed after gathering most of the requirements. At this stage the requirements were prioritized because it was not feasible to implement all the requirements within the targeted deadline for the project; the developer, advisors and sponsor made this decision jointly. The two major disadvantages of using CLI are listed in the following table:

Disadvantages	Solutions
Difficult to keep track what changed.	Version control allows comparison of 2 stored configurations and give users the ability to adequately comment on the different versions of configurations that they have created.
There are no built-in checks for semantic flaws, only syntax checking	Parsing existing configurations from network appliances, bringing to the user's attention any conflicts or redundancies in those configurations.

Table 1: Disadvantages of using CLI and solutions

3.2Functional requirements

Each functional requirement is given in the following format:

Index:

Name:

Priority:

Purpose:

Input parameters:

Action:

Output parameters:

Exceptions:

Remarks:

Index refers to a unique index assigned to this functional requirement. It will be used for cross-referencing this requirement from other requirements and also from other documents such as the design document.

Name is a descriptive name given to this functional requirement. This name does not need to be unique.

Priority informs the user of how critical the requirement is.

Purpose is a short description (in a line or two) of the function. It is used to quickly understand the functionality and is also used to search the required function when all the functionalities are browsed through.

Input parameters refer to a set of parameters that the given function accepts as input. These parameters are required in order to design and implement the current function. No type information will be included for parameters in this document.

Action refers to the action or function that this requirement should be performed given the input parameters and output parameters.

Output parameters refer to a set of parameters that are output/exhibit by the current function, when implemented. No type information will be included for parameters in this document.

Exceptions refer to a set of conditions, each of which indicates a situation in which the function will stop. Notice that this column only lists a set of exceptions that might occur but does not suggest any action that must be taken when the exceptions occur. These actions will be included in the design document.

Remarks include a set of comments that explain more about the functionality. It also describes hints to the designer and implementer that are suggested by the requirements analyst.

3.2.1 Initialized the System

Index: SYS.1

Name: InitializedSystem

Priority: High

Purpose: To initialize the System.

Inputs: None

Outputs: None

Actions:

- Check for the directory ".*savedConfigurations*\" exist or not.
- Create ".*savedConfigurations*\" directory to store the configuration if it is not exists in the system.

Exceptions: None

Remarks:

This function may be used whenever the System needs to be reset. This function is implemented at the initial stage of the software, the first time the software is installed.

3.2.2 Load System

Index: SYS.2

Name: LoadSystem

Priority: High

Purpose: To load the previous information stored in the system.

Inputs: List of configuration filenames stored in the system.

Outputs: None

Actions:

- Ensure the directory which stored configuration is exists.
- Load filename of all stored configuration filenames.

Exceptions:

- The directory that stored configuration is invalid (directory .*savedConfigurations*\ does not exist).

Remarks:

If there are no files in ".*savedConfigurations*\" (e.g. no configuration was stored), that is acceptable.

This function is same with function get list of stored configurations.

3.2.3 Retrieve configuration

Index: SYS.3

Name: RetrieveConfiguration

Priority: High

Purpose: To retrieve the configuration and description from remote Cisco appliance with IP address, Telnet password and Enable password.

Inputs: IP address, Telnet password, Enable password.

Outputs: The configuration of remote Cisco appliance.

Actions:

- Ensure IP address field is not empty.
- Ensure IP address exists in the network.
- Ensure Telnet password field is not empty.
- Ensure Enable password field is not empty.
- Ensure that Telnet password & Enable password are working with remote Cisco appliance at this IP address.
- Open Telnet connection to Cisco appliance
- Login to the device according to IP address.
- Retrieve the configuration
- Retrieve the description of the configuration
- Close Telnet connection to Cisco appliance

Exceptions:

- IP address field is empty.
- IP address is invalid (valid format is xxx.xxx.xxx.xxx, 0<xxx<256).
- The remote Cisco appliance does not exist.
- Telnet password field is empty.
- Enable password field is empty.
- Telnet password & Enable password do not work with the remote Cisco appliance.

Remarks:

To connect to remote Cisco appliance, the remote device must be configured to accept Telnet connection (see Appendix for necessary commands) and in the same network with the computer running this application.

The configuration of the Cisco device will retrieve by the command "*show running-config*". This command will return all configurations for specific router/firewall. This configuration may or may not same with the *startup-config*.

The description of the configuration will retrieve by the command "*show version*". It will be included the information about:

- Cisco software version
- Chassis type
- Bootstrap information
- Uptime
- Reload reason
- Reload time

- Software image file
- Memory information
- Flash information
- Configuration registry
- Password recovery mechanism

3.2.4 Upload access control list

Index: SYS.4

Name: UploadACLs

Priority: High

Purpose: To upload the access control list to remote host with IP address, Telnet password and Enable password.

Inputs: New access control list, Access control list name/number, IP address, Telnet password, Enable password.

Outputs: None

Actions:

- Ensure New access control list field is not empty.
- Ensure Access control list name/number is not empty.
- Ensure IP address field is not empty.
- Ensure IP address exists in the network.
- Ensure Telnet password field is not empty.
- Ensure Enable password field is not empty.

- Ensure that Telnet password & Enable password are working with remote Cisco appliance at this IP address.
- Open Telnet connection to Cisco appliance
- Login to the device according to IP address.
- Upload new access control list to the device
- Close Telnet connection to Cisco appliance

Exceptions:

- New access control list field is empty.
- Access control list name/number field is empty.
- IP address field is empty.
- IP address is invalid (valid format is xxx.xxx.xxx.xxx, 0<xxx<256).
- The remote Cisco appliance does not exist.
- Telnet password field is empty.
- Enable password field is empty.
- Telnet password & Enable password do not work with the remote Cisco appliance.

Remarks:

Because the application will be used on main campus network system, it is mandatory to maintain the stable of the core network. Any changes made to the configuration of Cisco device will affect immediately but it will be easily to roll-back to previous configuration if there is any issue with the new one. To permanently store the new configuration, the network administrator must

be manually login to the device and run the command "*copy running-config starup-config*" to store the configuration to flash memory of the device. It may be not convenience but it guarantee the core network will tolerate by any mistake when using the application.

3.2.5 Save configuration file

Index: SYS.5

Name: SaveConfigurationFile

Priority: High

Purpose: To save the configuration file for later use.

Inputs: Configuration of Cisco appliance

Outputs: None

Actions:

- Ensure the configuration is not empty.
- Store the configuration with format "*yyyy.mm.dd_hostIP_vhhmmss.txt*"

Exceptions:

- The configuration is empty.

Remarks:

All the files relative to the configuration will be store under directory "*.\savedConfigurations*".

3.2.6 Save description of selected configuration file

Index: SYS.6

Name: SaveDescriptionConfigurationFile

Priority: High

Purpose: To save the description of selected configuration file.

Inputs: Configuration filename, the description of the configuration filename

Outputs: None.

Actions:

- Ensure the configuration file name field is not empty.
- Ensure the configuration file is exists in the system.
- Ensure the description of the configuration is not empty.
- Store the description of the configuration with format "*yyyy.mm.dd_hostIP_vhhmmss.txt.description*"

Exceptions:

- The configuration file name field is empty.
- The configuration file does not exist in the system
- The description of the configuration is empty.

Remarks:

All the files relative to the configuration will be store under directory "*.\savedConfigurations*".

3.2.7 Delete configuration file

Index: SYS.7

Name: DeleteConfigurationFile

Priority: High

Purpose: To delete configuration file and its description.

Inputs: Configuration filename.

Outputs: None

Actions:

- Ensure configuration file exists in the system.
- Ensure the description of the configuration file exists in the system.
- Delete the configuration file.
- Delete the description of the configuration file.

Exceptions:

- The configuration file does not exist in the system.
- The description of the configuration file does not exist in the system.

Remarks:

Be carefully to delete the configuration because the application has not required implementing the function to revert the deleting action.

3.2.8 Get list of stored configurations

Index: SYS.8

Name: GetListStoredConfigurations

Priority: High

Purpose: To get list of stored configurations which user saved.

Inputs: None.

Outputs: List of stored configuration files.

Actions:

- Ensure the directory ".*savedConfigurations*\" is correct.
- Get list of all files ending with ".txt" in this directory.

Exceptions:

- The directory ".*savedConfigurations*\" does not exist.

Remarks:

If there is no configuration file in this directory, it is acceptable.

Default directory is working directory + "\savedConfigurations\".

The format of configuration file is yyyy.mm.dd_hostIP_vhhmmss.txt

3.2.9 Get configuration of selected configuration file:

Index: SYS.9

Name: GetConfigurationSelectedConfigurationFile

Priority: High

Purpose: To view configuration of selected configuration file from the list.

Inputs: Configuration filename

Outputs: Configuration details stored in the file.

Actions:

- Ensure the filename field is not empty.
- Ensure the file is exists in the system.
- Get the content of the configuration file.

Exceptions:

- The configuration filename field is empty
- The configuration file does not exist in system.

Remarks:

Configuration filename is the plain text file with format
"*yyyy.mm.dd_hostIP_vhhmmss.txt*"

3.2.10 Get description of selected configuration file:

Index: SYS.10

Name: GetDescriptionSelectedConfigurationFile

Priority: High

Purpose: To get the description of selected configuration file from the list.

Inputs: Configuration filename

Outputs: Description of the configuration file.

Actions:

- Ensure the filename field is not empty.
- Ensure the file is exists in the system.
- Ensure the description file is exists in the system.
- Get the content of the description of the configuration file.

Exceptions:

- The configuration filename field is empty
- The configuration filename does not exist in system.
- The description filename does not exist in system.

Remarks:

Description file is the file with format "*yyyy.mm.dd_hostIP_vhhmmss.txt.description*" which is stored in the same directory with configuration files.

3.2.11 Compare 2 stored configuration files:

Index: SYS.11

Name: Compare2ConfigurationFiles

Priority: High

Purpose: To compare two configuration files which stored in the system.

Inputs: Configuration filename 1, Configuration filename 2

Outputs: Differences between two configuration files.

Actions:

- Ensure filename 1 field is not empty.
- Ensure file 1 is exists in the system.
- Ensure filename 2 field is not empty.
- Ensure file 2 is exists in the system.
- Compare and highlight the differences between two configuration files.

Exceptions:

- The configuration filename 1 field is empty
- The configuration file 1 does not exist in system.
- The configuration filename 2 field is empty
- The configuration file 2 does not exist in system.

Remarks:

Two configuration files should be retrieved from same Cisco appliance.

3.2.12 View list access lists

Index: SYS.12

Name: ViewListACLs

Priority: High

Purpose: To view all access lists of selected configuration.

Inputs: Configuration filename

Outputs: List of access lists of selected configuration.

Actions:

- Ensure the configuration filename is not empty.
- Ensure the configuration file is exists in the system.
- Get list all access lists in the configuration file.

Exceptions:

- The configuration filename field is empty.
- The configuration file does not exist.

Remarks:

The access list number is a string start with "*ip access-group* …" in the configuration.

3.2.13 View list routes

Index: SYS.13

Name: ViewListRoutes

Priority: High

Purpose: To view all routes in the configuration.

Inputs: Configuration filename

Outputs: All routes in the configuration.

Actions:

- Ensure the configuration filename is not empty.
- Ensure the configuration file is exists in the system.
- Get list all routes in the configuration file.

Exceptions:

- The configuration filename field is empty.
- The configuration file does not exist.

Remarks:

The route is a command line start with "*ip route …*".

3.2.14 View list interfaces

Index: SYS.14

Name: ViewListInterfaces

Priority: High

Purpose: To view all interfaces in the configuration.

Inputs: Configuration filename

Outputs: All interfaces in the configuration.

Actions:

- Ensure the configuration filename is not empty.
- Ensure the configuration file is exists in the system.
- Get list all interfaces in the configuration.

Exceptions:

- The configuration filename field is empty.
- The configuration file does not exist.

Remarks:

The interface is a command line start with "*interface ...*".

3.2.15 View details of selected interface configuration

Index: SYS.15

Name: ViewInterfaceConfiguration

Priority: High

Purpose: To view all configuration of selected interface.

Inputs: Configuration, interface name

Outputs: The configuration of selected interface.

Actions:

- Ensure the configuration filename is not empty.
- Ensure the configuration file is exists in the system.
- Ensure the interface name is not empty.
- Get the configuration details of the interface in the configuration file.

Exceptions:

- The configuration filename field is empty.
- The configuration file does not exist.
- The interface name field is empty.

Remarks:

The configuration details of the interface are a set of command lines between interface name and character "!".

3.2.16 View list access lists of selected interface

Index: SYS.16

Name: ViewListACLsSelectedInterface

Priority: High

Purpose: To view list all access lists of selected interface.

Inputs: Interface configuration

Outputs: List of access lists name/number of selected interface.

Actions:

- Ensure the configuration details of the selected interface are not empty.
- Get all access lists name/number in the configuration details of the selected interface.

Exceptions:

- The configuration details of the selected interface are empty.

Remarks:

The access list number is a string start with "*ip access-group ...*".

3.2.17 View access list details

Index: SYS.17

Name: ViewACLDetails

Priority: High

Purpose: To view access list details of selected access list name/number.

Inputs: Configuration filename, access list name/number

Outputs: Access list details of selected access list number.

Actions:

- Ensure the configuration filename is not empty.
- Ensure the configuration file is exists in the system.
- Ensure the access list name/number is not empty.
- Get access list details in the configuration file.

Exceptions:

- The configuration filename field is empty.
- The configuration file does not exist.
- The access list name/number field is empty.
- The access list name/number does not exist.

Remarks:

The access list detail is a command line start with "*access-list access-list-name/number*".

3.2.18 Analyze

Index: SYS.18

Name: Analze

Priority: High

Purpose: To analyze the access list details for any conflicts or redundancies.

Inputs: Access list details

Outputs: Any conflicts or redundancies.

Actions:

- Ensure access list details are exists in the system.
- Parse configuration for any conflicts or redundancies.

Exceptions:

- The access list details field is empty.
- The access list details do not exist in the system.

Remarks:

The access list details are only for same access list name/number.

3.3Design

This part will explain *how* the product is to achieve all the requirements.

3.3.1 High-Level architectural design

The architectural design is also known as *general design*. In this, a modular decomposition of the application is presented in Figures 2.

In overall view the network administrator uses this stand-alone application on his personal computer to:

- Get/save the configuration from/to the file on the local hard drive
- Retrieve/upload the configuration from/to online Cisco appliances.

The configuration after retrieval from the file or retrieve from online Cisco appliance will be processed/analyzed.

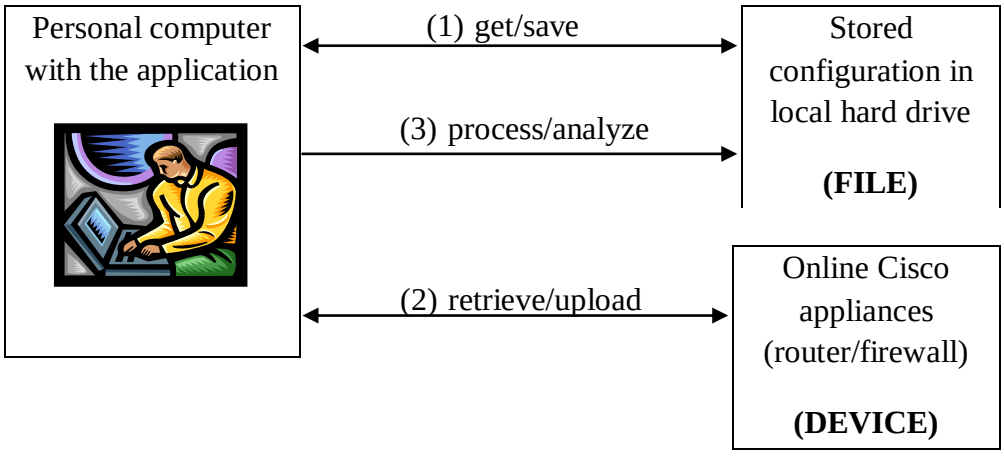


Figure 2: High Level Architecture of the application

As shown in the Figure 2, the application will always save the configuration to the file on the local hard drive after it is retrieved from the Cisco device. The configuration from the stored file will be used to process/analyze.

3.3.2 Detailed design

This design is also known as modular design. Here each of the modules is designed in detail.

Objects	Parameters	Methods
FILE	Filename	isSavedConfigurationDirExist() isStoredConfigurationExist() getListConfigurationFilenames() saveConfiguration() saveDescription() deleteConfiguration() getConfiguration() getDescription() compare(*filename1*, *filename2*) viewListACLs() viewListRoutes() viewListInterfaces() viewInterfaceDetails(*interface_name*) viewListACLsSelectedInterface(*interface_name*) viewACLDetails(ACL_Name) analyze()

DEVICE	IP_Address Telnet_Password Enable_Password	retrieve() upload(*ACL_Detail, ACL_Name*)

Table 2: Detailed design

3.3.3 UML Use case diagram

We can group the functions of this software into 3 parts:

- Interaction between network administrator and Cisco appliances (see Figure 3)

- Interaction between network administrator and stored configurations (see Figure 4)

- ACL parsing (see Figure 5)

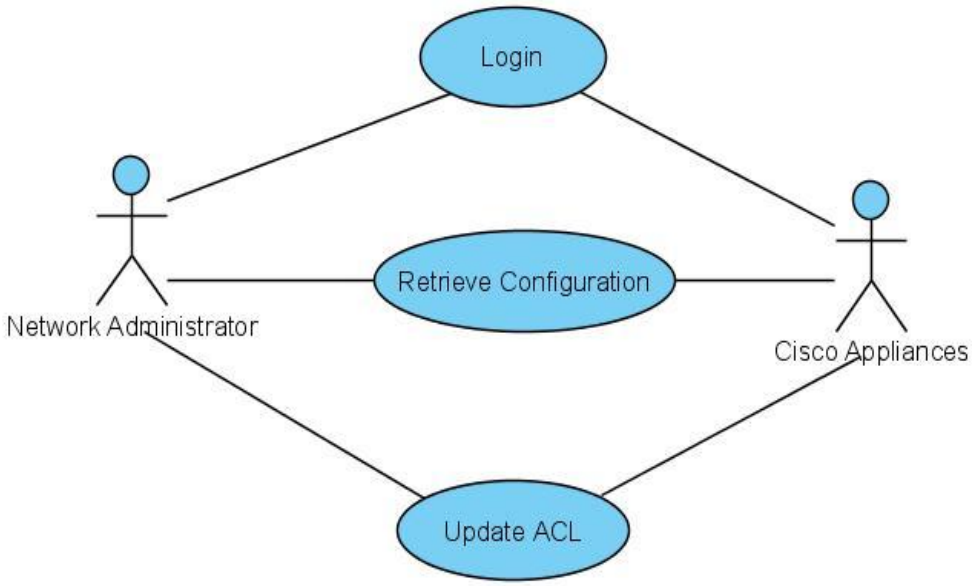


Figure 3: Interactive between network administrator and Cisco appliances

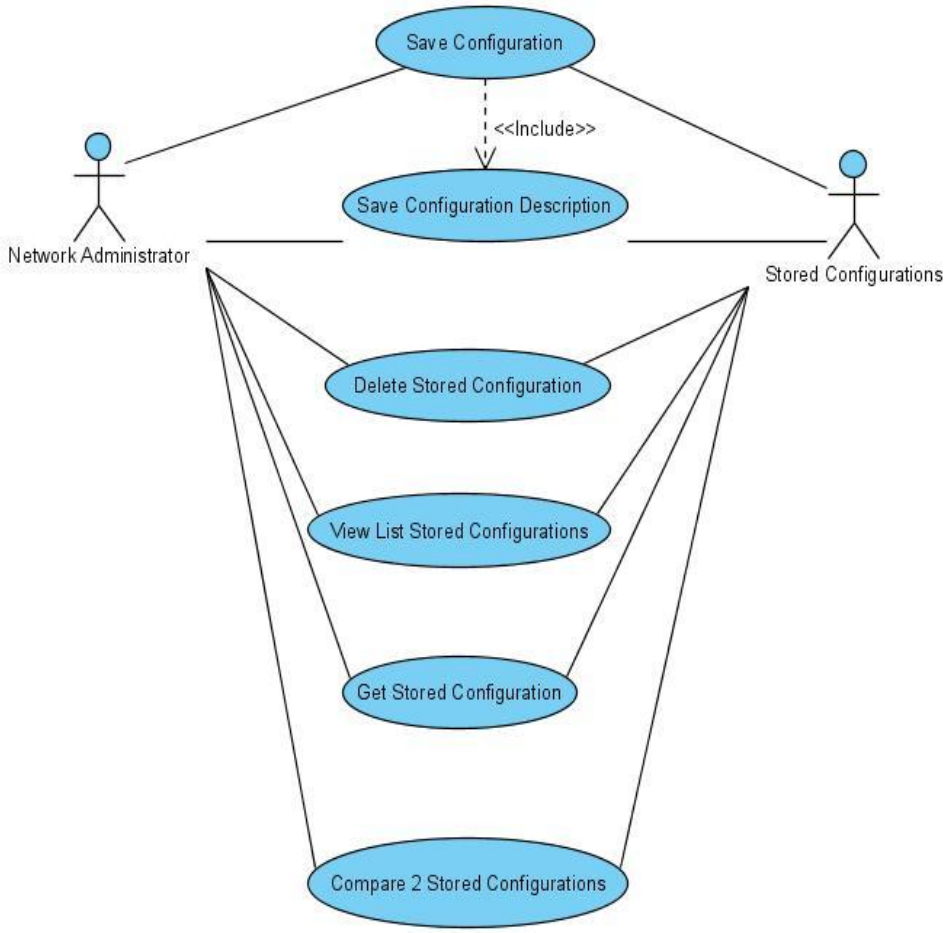


Figure 4: Interactive between network administrator and stored configurations

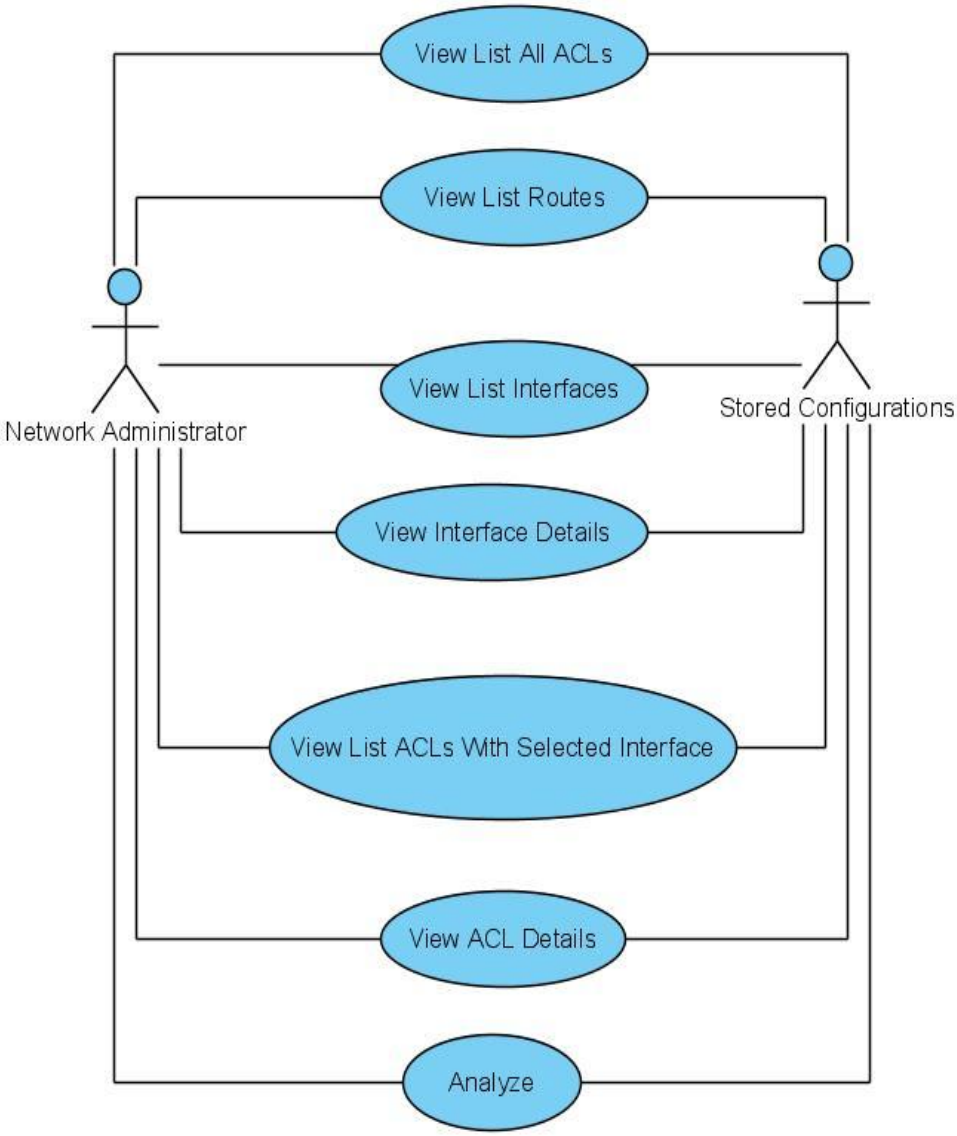


Figure 5: Useful tool to parse the ACL

3.3.4 Database design

There is no database used in this program. All of configuration will be stored on local hard drive with plain text format. Table 3 shows the format of files stored on the local hard drive and Table 4 shows where the files are stored:

File type	Filename format
Configuration	yyyy.mm.dd_hostIP_vhhmmss.txt
Description of configuration	yyyy.mm.dd_hostIP_vhhmmss.txt.description
Well-know ports	wellknowPorts.txt

Table 3: Format of stored file

File type	Directory
Configuration	.\savedConfigurations\
Description of configuration	.\savedConfigurations\
Well-know ports	.\conf\

Table 4: Format of directory

3.3.5 Security design

Security is not important in this project because Cisco devices provided log-in support, and all software can run alone on the administrator machine. It is safe to store the configuration in plain text format.

Telnet passwords and enable passwords are never stored anywhere on the machine.

3.3.6 User interface design

The project sponsor wanted to implement the parsing tool as stand-alone application for the following reason:

- Security risks: the sensitive information which retrieved from actual router/firewall should not be stored in public server where everyone can have access to it. Also bugs or misconfiguration problems in the server will allow hacker to access the content in that server. The stand-alone application storing information on the personal computer is a best solution for this application.
- End-user: a web-based application generally useful to use for a group of users. The user for this application is a sole network administrator. So it is not necessary to develop web application in this case.
- Offline working is possible.
- Centralized data is easy to backup.
- Updates can be made quickly and easily.

3.3.7 Windows design

Unlike general application where each window associates with one function, this application will have some additional windows and one primary main window with a lot of list boxes. The goal of this design is for these list boxes to display a part of configuration base under certain conditions. This is of benefit because the configuration tends to be very long and complex. Normal window design would require users to switch over many windows to see the desired information.

3.3.8 Language selected

This program needs to run on Windows, Linux, and MAC, so Java was chosen to develop this software. One of the most popular IDEs for developing Java applications is Eclipse, which was chosen to use in this project.

4. THE SOFTWARE APPLICATION

This program is designed to have three main functions allowing a network administrator to control Cisco appliances:

1) The application provides for remote access, permitting the retrieval of a configuration through network connection with **telnet** command.

2) The application parses configurations to identify redundancies and/or conflict in access lists.

3) The application stores different versions of configurations, allowing a network administrator to compare two stored versions and highlight the difference between two versions.

There are 16 modules were created for above functions:

4.1 Login

This software does not require username and password. However, the user must know the IP address and password to access remote device. If not, a user can only process stored configurations. Figure 6 shows the login panel. Fields may be left blank if no appliance access is required.

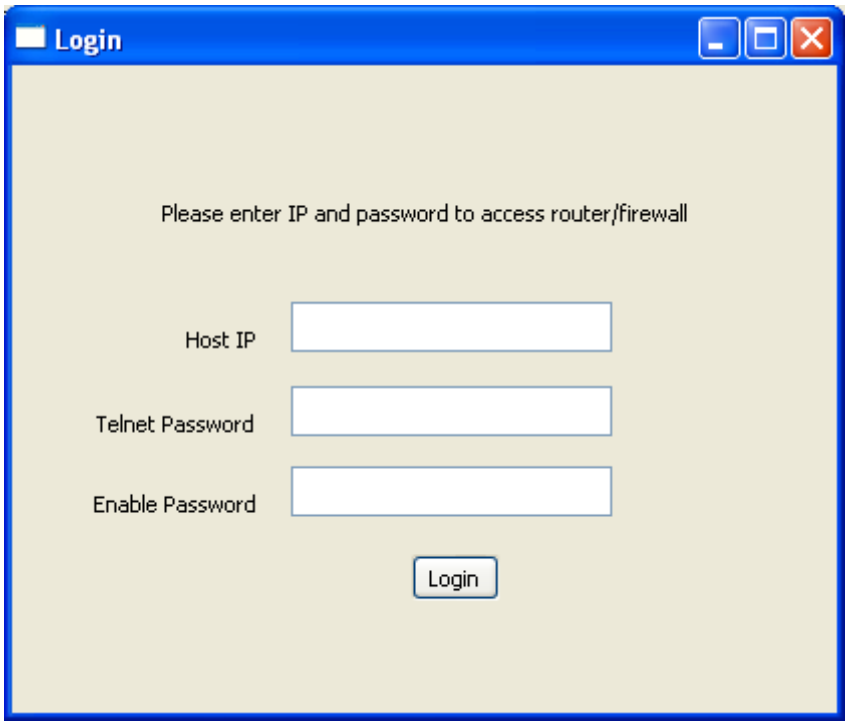


Figure 6: Login window

After click “Login” button, the main window will display. Figure 7 shows the main window.

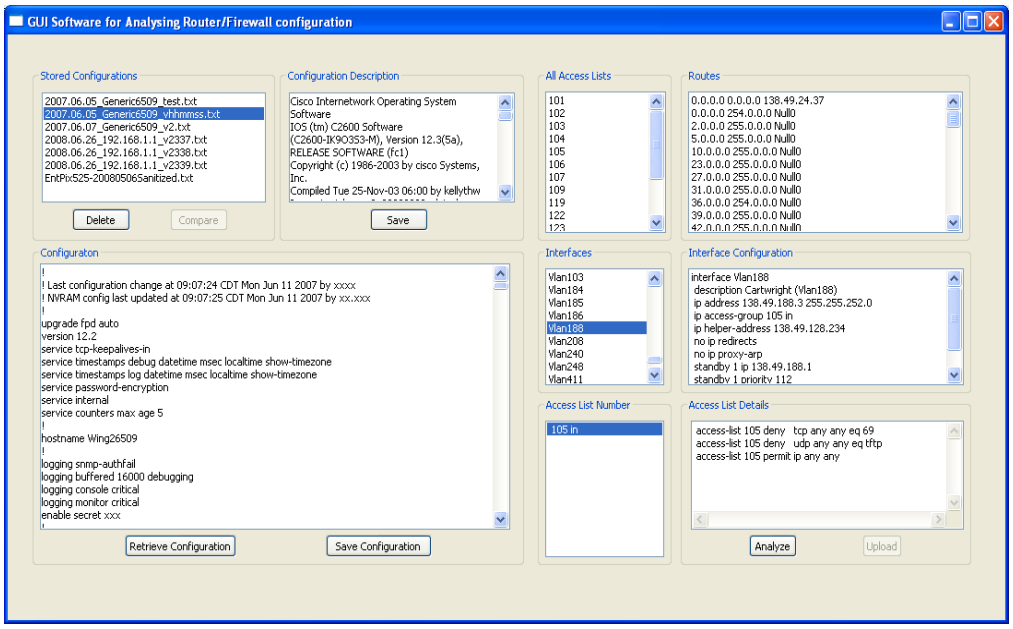


Figure 7: Main window

4.2 Retrieve Configuration

After a user clicks “Retrieve Configuration” button (from the main window), the IP address and the passwords from the login panel will be used to connect to a router/firewall by telnet command. Configuration details will be extracted from the device and stored to configuration file name “retrieved.txt”. The user can select the file name to show the content on the Configuration window. Figure 8 illustrates.

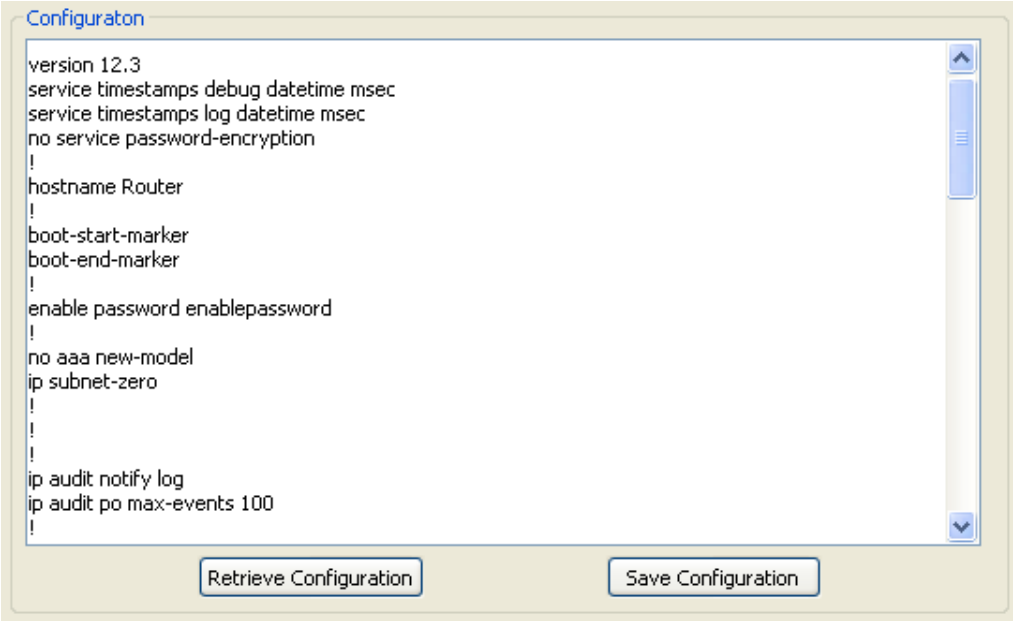


Figure 8: Configuration retrieves

When a configuration has been retrieved, the version also retrieves and stores this configuration description in a file named “retrieved.txt.description”. Figure 9 shows a sample popup window associated with a configuration description.

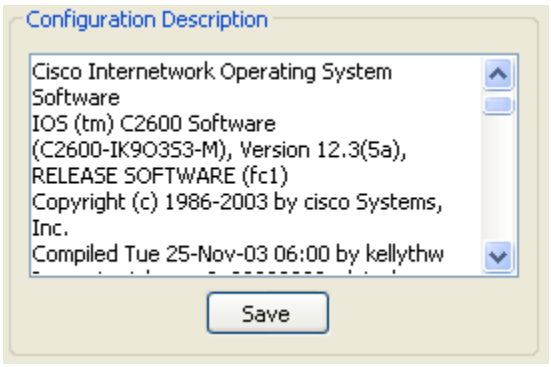


Figure 9: Configuration description

4.3 Upload ACL

This software allows a network administrator to upload separate ACLs (not necessarily the entire configuration) for security purposes and in case there is anything wrong with the configuration. Editing a configuration will not automatically update the appliance. To do that, the software allows the network administrator to select a specific interface to retrieve the ACLs name, or the administrator can select the ACLs name directly from the list. Figure 10 illustrates.

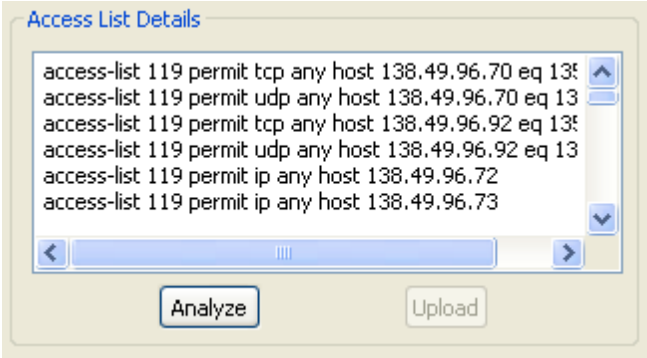


Figure 10: Upload ACL

This function of selecting an ACL is only active when working with retrieved configuration. The working configuration is “retrieved.txt”.

4.4 Save configuration

At anytime, a user can click “Save Configuration” to store the configuration and description. If the user is working with previous stored configuration, the file name will not change. Otherwise, the new file name will be automatically created and stored on local hard drive. Users do not need to store configuration descriptions separately.

New file name format:

.\savedConfigurations*yyyy.mm.dd_loginIP_*vhhmmss.txt

4.5 Save Configuration Description

After select configuration file name to work with, a user can click “Save” to store the description of the configuration.

4.6 Delete Stored Configuration

Stored configuration and description can be removed anytime without restriction. Users click “Delete” button to remove it. This button is only enable when a user selects a configuration file name.

4.7 View List Stored Configurations

All stored configuration will be shown in this window. Figure 11 demonstrates.

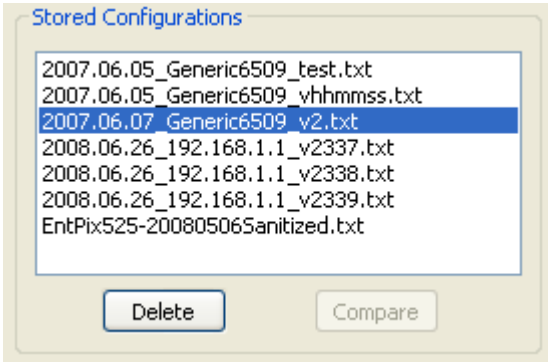


Figure 11: Stored Configurations window

Users can select one file name to activate the “Delete” button. To activate the “Compare” button, a user must select 2 file names.

4.8 Get Stored Configurations

When a file name is selected from the list of stored configurations, the configuration and the description will be shown in the appropriate window. Also, all access lists, routes, and interfaces will be parsed and shown in these windows. Figure 12 shows a sample of the main window after storing configurations.

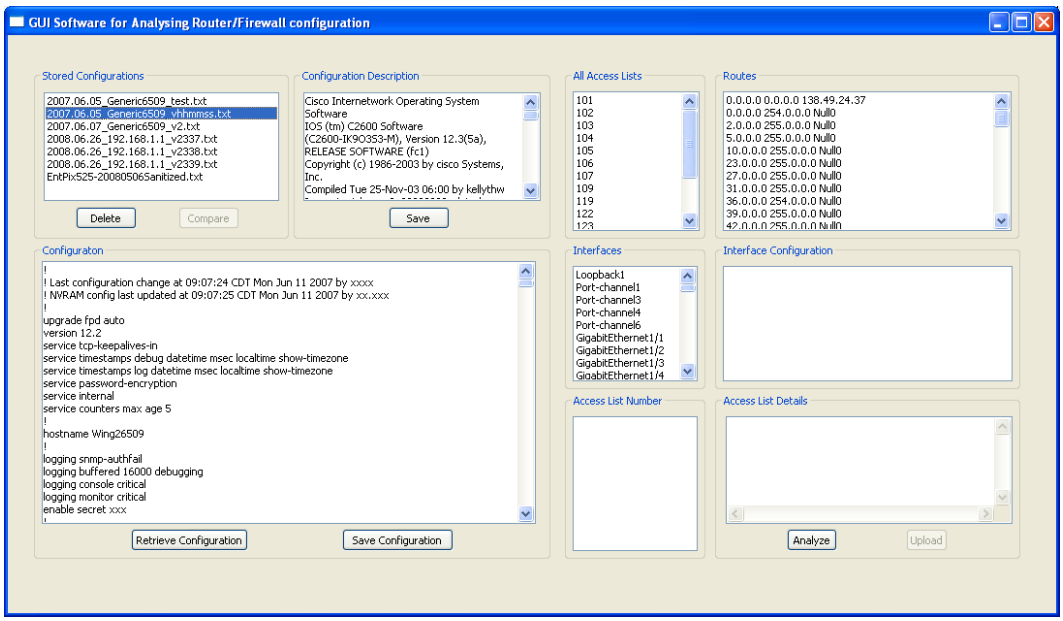


Figure 12: Get stored configurations

4.9 Compare 2 Stored Configurations

A network administrator can compare two configurations by choosing the configurations from the list, then clicking the “Compare” button. If there are two file names chosen, the “Compare” button is activate and the “Delete” button will be de-active. After clicking “Compare” a new window (like the one in Figure 13) will open with the different parts highlighted.

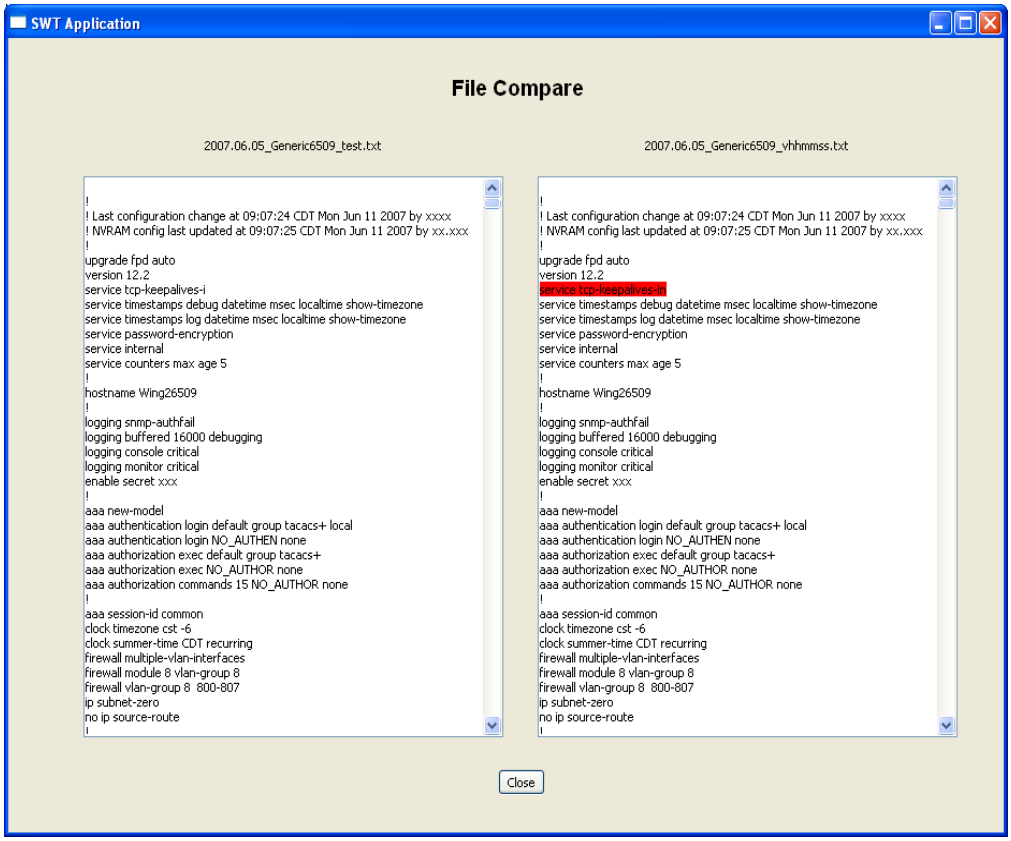


Figure 13: File compare

This function will compare two configurations line by line. Every line in one configuration which is not in the other configuration will be highlighted. Users can scroll the vertical bar of any two windows, the other windows will automatic scroll at the same speed.

If a router configuration is updated frequently, the configuration tends to become longer and longer. The comparison function and associated versioning ability should prove helpful. Using this function, a network administrator can know what has been added, changed or removed from the old version. In fact the whole configuration is compared; however, only specific ACLs are uploaded to the router during update and/or restoration.

4.10 View List All ACLs

When user selects stored configuration filename, this function will automatically start to get the list of all access lists in the configuration and show in All Access Lists window as shown in Figure 14.

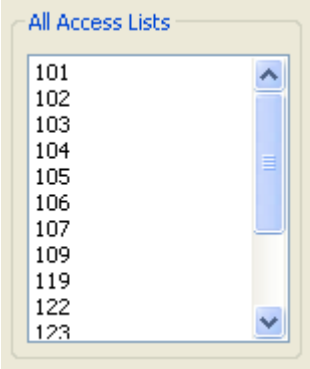


Figure 14: All Access Lists window

If user selects the access list, the access list detail will show up in the Access List Details window.

4.11 View List Routes

When a user selects stored configuration filename, this function will automatically start to get the list of all routes in the configuration and display them in the Routes window (see Figure 15).

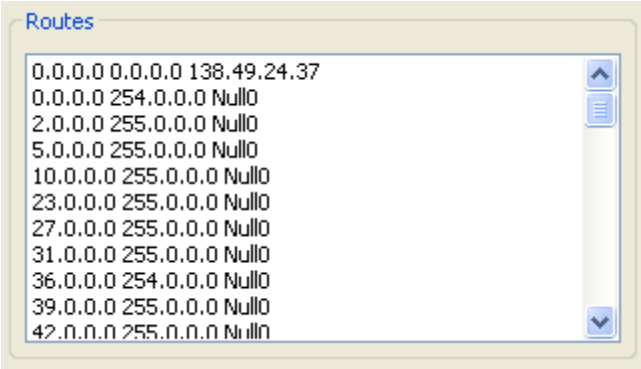


Figure 15: Routes window

These routes will be compared with access list. If a destination of access list match with the route commands, that should be a potential Routes/ACL

conflict. In this case, the software will not analyze the access list. A Route/ACL conflict is reported by an associated popup window like the one shown in Figure 16.

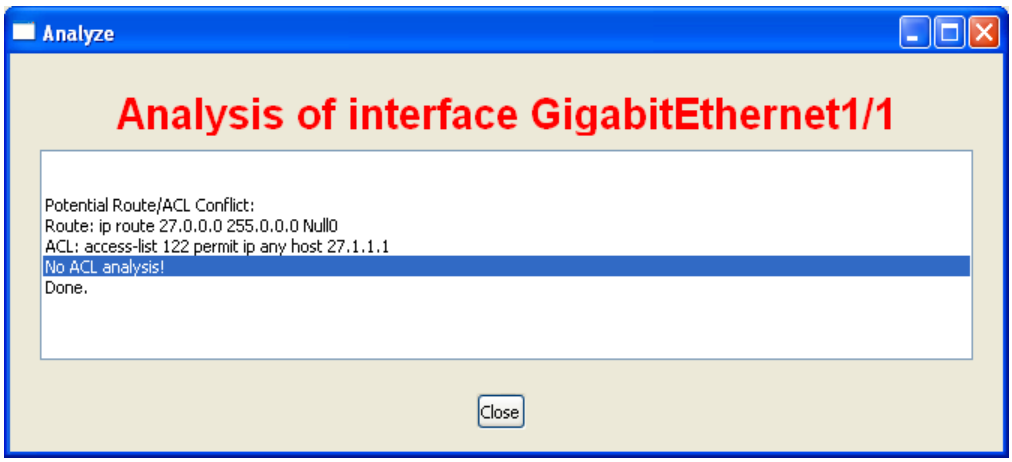


Figure 16: Route/ACL conflict

4.12 View List Interfaces

When user selects stored configuration filename, this function will automatically start to get the list of all interfaces in the configuration and show in Interfaces window as shown in Figure 17.

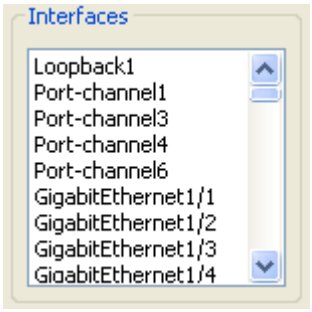


Figure 17: Interfaces window

4.13 View Interface Details

When a user selects an interface in Interfaces window, this function will get the configuration of that interface and show in Interface Configuration window (see Figure 18).

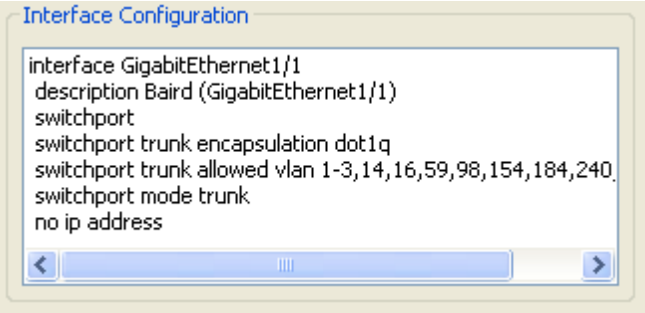


Figure 18: Interface Configuration window

4.14 View List ACLs with Selected Interface

When a user selects an interface in Interfaces window, this function will parse the configuration to get list of all access lists assigned with that interface (if any) and display in Access List Number window as illustrated in Figure 19.

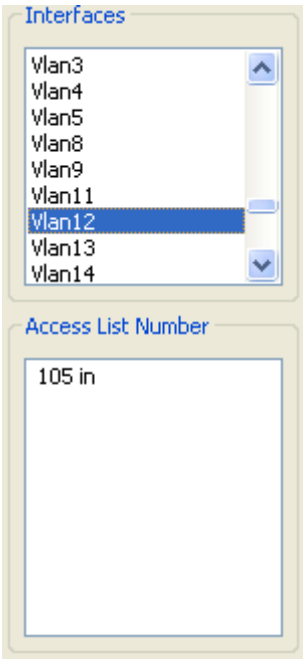


Figure 19: Access List Number window with a selected interface

4.15 View ACL Details

When a user selects an access list in All Access Lists window or Access List Number window, this function will parse the configuration to get the configuration of that access list and display in the Access List Details window. Figure demonstrates.

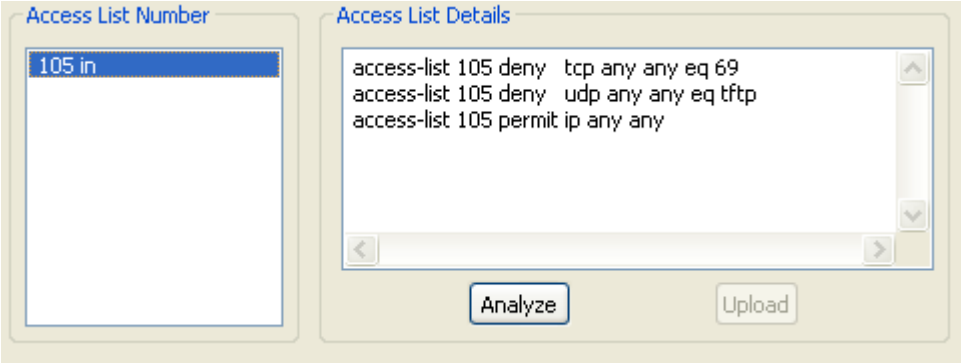


Figure 20: Access List Details window

From this window, a user can edit the access list directly and upload to the router. The user will still need to use command "*copy running-config startup-config*" on router to store the access list. This is done to provide for system stability; if there is something wrong, a user can restart the router/firewall to reverse the change.

4.16 Analyze

4.16.1 Well-known ports

Before the program can analyze the access list, all access lists need to be converted to a standard format which can be understand by the program. In standard format, all well-known ports need to be converting to port numbers. This can be done by specific port name and port number in file "wellknowPorts.txt"

Example:

wellknowPorts.txt

http 80

telnet 23

ftp 21

4.16.2 Object-group and names

In a firewall configuration object-group and names are used widely for ease of creating configurations.

For example:

```
names
name238.29.28.12 OraApp1
name 238.29.32.10 sandypc
name 238.29.42.20 johnpc

object-group network RemoteDesktop
description Access servers behind firewall with Remote Desktop
network-object sandypc255.255.255.255
network-object johnpc255.255.255.255

object-group service People-wsdev01 tcp
description People Installation for wsdev01
port-object eq 40300

access-list inside_access_in permit tcp host OraApp1 object-group RemoteDesktop object-group
People-wsdev01
```

This should be convert 2 standard access list:

```
access-list inside_access_in permit tcp host 238.29.28.12 host 238.29.32.10 eq 40300

access-list inside_access_in permit tcp host 238.29.28.12 host 238.29.42.20 eq 40300
```

4.16.3 How to compare two ACLs commands

The application is designed to match all combination of ACL and generate the notification when redundancies or conflicts are found.

The key to compare two ACL commands is to convert the range of IP addresses and IP address to one standard format. This conversion is based on the fact that wildcard mask bits specify which bits can change and which cannot. If a bit of an IP address cannot change, we simple do not care about it and mark that bit as 'x' (See Table 5).

	Decimal form	**Binary**
Wildcard mask	0.0.0.255	00000000.00000000.00000000.**11111111**
IP address	192.168.1.0	11000000.10101000.00000001.**00000000**
Standard format		11000000.10101000.00000001.**xxxxxxxx**

Table 5: Converting range of IP addresses to standard format

	Decimal form	**Binary**
Wildcard mask	0.0.0.0[4]	00000000. 00000000. 00000000. 00000000
IP address	192.168.1.2	11000000.10101000.00000001.00000010
Standard format		11000000.10101000.00000001.00000010

Table 6: Converting IP address to standard format

[4] We assign wildcard mask is 0.0.0.0 for host address, mean all bit cannot changed

Now what we need to do is compare two results:

Result One = **11000000.10101000.00000001**.xxxxxxxx

Result Two = **11000000.10101000.00000001**.00000010

We can easily see that first 24 bits are same between two results. That mean, two ACLs have the address overlaps. Figure 21 illustrates the associated popup window reporting the conflict and Figure 22 shows a redundancy support.

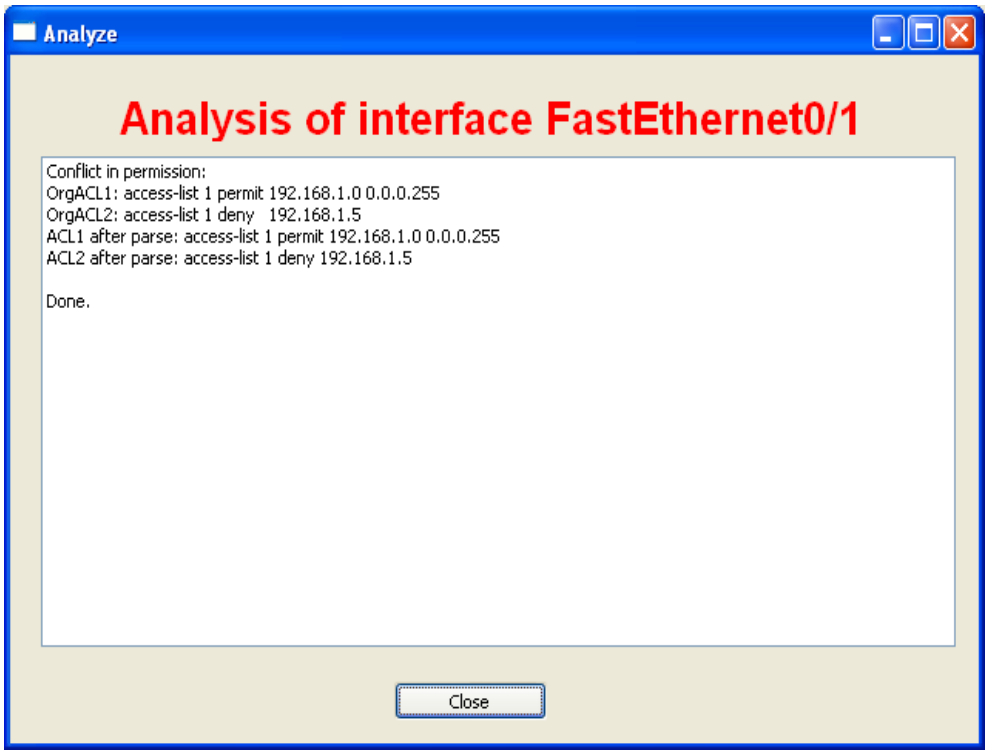


Figure 21: Conflict in two access lists

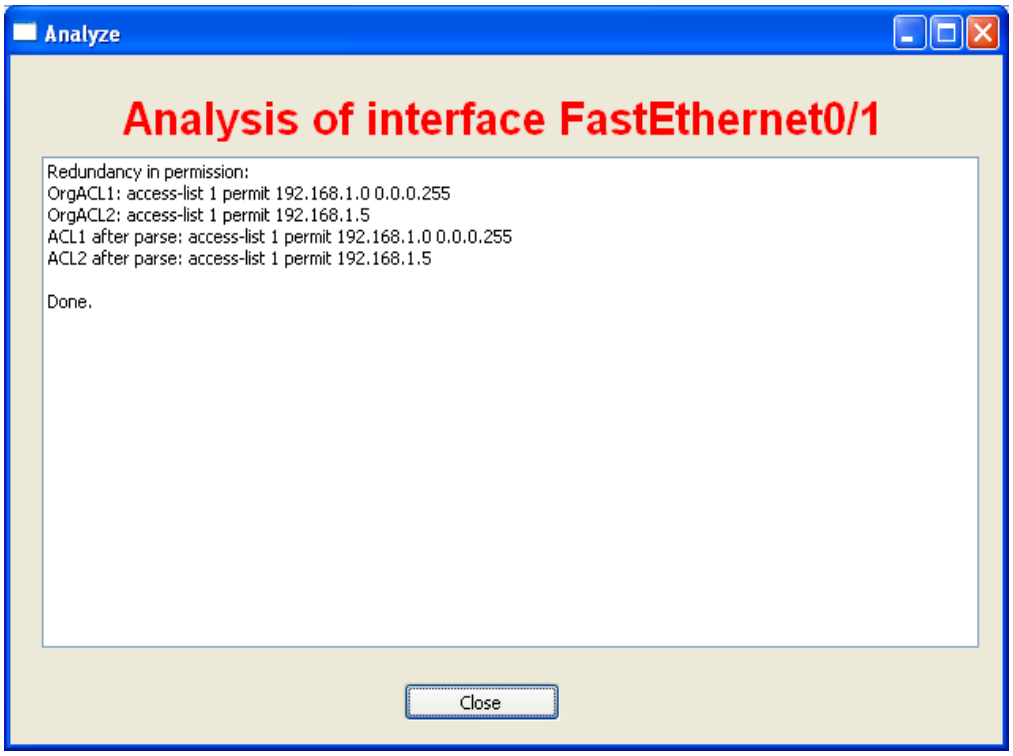


Figure 22: Redundancy in two access lists

5. TESTING

Testing itself cannot be expected to catch all errors in the application. Multiple testing phases and methods were used in this program to improve the testing effectivenes.

5.1 Requirement analysis

The requirement document was given to the project sponsor to make sure that all the requirements were captured in the document. After several meetings, the requirement document was updated to include all requirements.

5.2 Design analysis

The application was design through two steps:

- High level design: the complete product is divided into components
- Detail design: these components in turn are designed independently of the fact that it is to be a component of the product as a whole.

This design will make sure that these components will fit together correctly and the whole product will do what it is supposed to do.

5.3 Implementation testing

To guarantee the application cans perform correctly in using, it has been tested throughout several processes when the program was developed.

These testing steps have been applied to this application:

- Unit testing: this method of testing verified that individual units (method) of source code are working properly, reduced false positives and increased the effectiveness of the test suite. Unit test should eliminate uncertainty in the method itself and help the integration process avoid unpredictable exception. Each method of the program is isolated and tested with test cases that cover all paths through the method, so that the fault can be quickly identified and fixed. However, this test may not able to catch integration error or other system-wide issues.

- Integration testing: all software methods were combined and tested as a group. Integration test exposes defects in the interfaces and interaction between integrated methods and classes. The group of methods and classes were exercised through their interface using black box testing. Test cases were created to test whether or not group of methods and classes interact correctly.

- System testing: the application was tested with a set of test cases and real configurations from using Cisco appliances in core network to verify that it meets its requirements using black box testing.

6. FUTURE WORK

The real-world network is a set of many network appliances connected to form a network. It is necessary to draw a map of network and trace the data from the start point to the end point.

7. CONCLUSION

This software is a useful tool for analyzing router and firewall configurations. The management of configuration files which used to difficult, time-consuming, and error-prone (because the files tend to be large and complex, especially when they contain large numbers of ACLs) is become a few clicks with high accuracy (up to 100%).

Core functions:

- Login

- Retrieve Configuration

- Upload ACL

- Save Configuration

- Save Configuration Description

- Delete Stored Configuration

- View List Stored Configuration

- Get Stored Configuration

- Compare 2 Stored Configurations

- View List All ACLs

- View List Routes

- View List Interfaces

- View Interface Details

- View List ACLs with Selected Interface

- View ACL Details

- Analyze

APPENDIX

Pre-configuration:

Before the router/firewall can be used with this software, it must be pre-configure by network administrator through console port. This is necessary to allow access from the network to interface and telnet service enabled on the device:

- The Ethernet connection between the router and the workstation must be made with a crossover cable.

- The workstation must be on the same subnet as the router's Ethernet interface.

These commands below are necessary:

enable password xxxx

config terminal

line vty 0 4

password xxx
login

Assumption

- This software is independent and totally self-contained.

- The product is a desktop application and will be used by the network administrator of router or switch.

- The user must be a Cisco network administrator to use this software.

- This software is not providing function to recover password.

- This software is not providing function to find existing device on the network.

- This software is not generating configuration by itself.

- Any change in the configuration by software GUI will not affect the remote device until the user uploads it.

- Only one device can be managed at a time.

- Each remote device has to have unique IP address. Otherwise, the user may be upload wrong configuration.

BIBLIOGRAPHY

1. John Albritton, *Cisco IOS Essentials*, McGraw-Hill, 1999.
2. Kenneth L. Calvert, Michael J. Donahoo,*TCP/IP Sockets in Java: Practical Guide for Programmers*, Morgan Kaufmann Publishers, 2002.
3. Sackett,*Cisco Router Handbook*,McGraw-Hill, 1998.
4. Herbert Schildt, *Java: The Complete Reference, J2SE 5 Edition*, McGraw-Hill, 2004.
5. James Boney, *Cisco IOS in a Nutshell (2nd Edition)*, O'Reilly, 2005.
6. Jeff Sedayao, *Cisco IOS Access Lists*, O'Reilly, 2001.
7. Todd Lammle, *Cisco Certified Network Associate*, Wiley Publishing, 2007.
8. I. Sommerville, *Software Engineering 6th Ed.*, Addison Wesley, 2001.
9. A.M. Davis, *Software Requirements Analysis and Specification*, Prentice Hall, 1990.
10. Computer Society/Software and Systems Engineering Standards Committee, "*Recommended Practice for Software Requirements Specifications*", Std. 830-1998, IEEE Standards Association, Piscataway, NJ, September 16, 1997.
11. Hans van Vliet, *Software Engineering: Principles and Practice*, John Wiley & Sons, Ltd., 2000.
12. Stephen R. Schach, *Practical Software Engineering*, Aksen Associates Incorporated Publishers, 1992.
13. Stephen R. Schach, *Software Engineering*, Aksen Associates Incorporated Publishers, 1990.
14. Penny A. Kendall, *Introduction to Systems Analysis and Design: A Structured Approach, 2.nd Edition*, Wm. C. Brown Publishers, 1992.