Software Requirements Specification

for

Real-time Network Analysis

Version 1.0

Prepared by Michael King

Louisiana Tech University

3 May 2010

Table of Contents

List of Tables

List of Figures

Revision History

Name	Date	Reason For Changes	Version
Michael King	24 Mar 2010	Initial revision	0.5
Michael King	16 Apr 2010	Added Development Cycle, updated information	0.6
Michael King	30 Apr 2010	Added System Features, more updates	0.7
Michael King	3 May 2010	Added Final details, charts. Fixed format errors	1.0

1. Introduction

1.1 Purpose

This document sets forth the software requirements for the Real-time Network Analysis (ReTiNA) system revision 1. This system is to be used as a sub-system within the Cyberstorm Infrastructure and Scoring system. This document also covers the design phase of the ReTiNA system and an explanation of the various module's functions and communication system.

1.2 Document Conventions

In this document, the term "Cyberstorm system" refers to the complete system as designed and developed by the CSC404 Capstone class from Louisiana Tech University in the Spring of 2010. "System" refers to the ReTiNA software system, which is a component of the Cyberstorm system. "Module" refers to one of the five components that make up the ReTiNA system.

All listed requirements are assumed to be a component of the ReTiNA system unless stated otherwise, with the exception of the System Features (5) section, where any stated requirements will be assumed to belong to the component being referenced.

1.3 Intended Audience and Reading Suggestions

The intended audience categories for this document are project managers, technical users, developers, and purchasing agents including executive officers. It is recommended that all readers read the Introduction (1).

- Project managers will be particularly interested in the Overall Description (2) and External Interface Requirements (3) to ensure the system can integrate with existing infrastructure.
- Technical Users includes the system administrators who will actually be working with the system on a day to day basis. These users should pay particular attention to the External Interface Requirements (3), System Features (5), and Other Nonfunctional Requirements (6).
- Developers would benefit most from the Development Cycle (4) which covers the process behind designing and implementing the system.
- Purchasing agents will be able to gain a working knowledge of the systems capabilities from the Overall Description (2).

1.4 Product Scope

The purpose for ReTiNA is to reside on a network router and watch the packets passing through. The system can detect harmful packets and provide warnings similar to an IDS. It is also be capable of detecting what machines are sending information through the network. This will allow an observer to see who is on a network and what attacks are currently taking place. Finally, the system is capable of performing node detection to determine when a new computer has begun sending information over the network. ReTiNA allows a systems administrator to monitor the activity on the network in real time and provide timely response to any intrusion attempts.

2. Overall Description

2.1 Product Perspective

The ReTiNA system is a relatively unique product designed to function as a part of the Cyberstorm system. The system provides a real-time situational awareness of a network, with the primary requirement of relaying network and attack activity to spectators at the Cyberstorm event. Figure 1 shows a Data Flow Diagram for the Cyberstorm system and how the ReTiNA system is related.

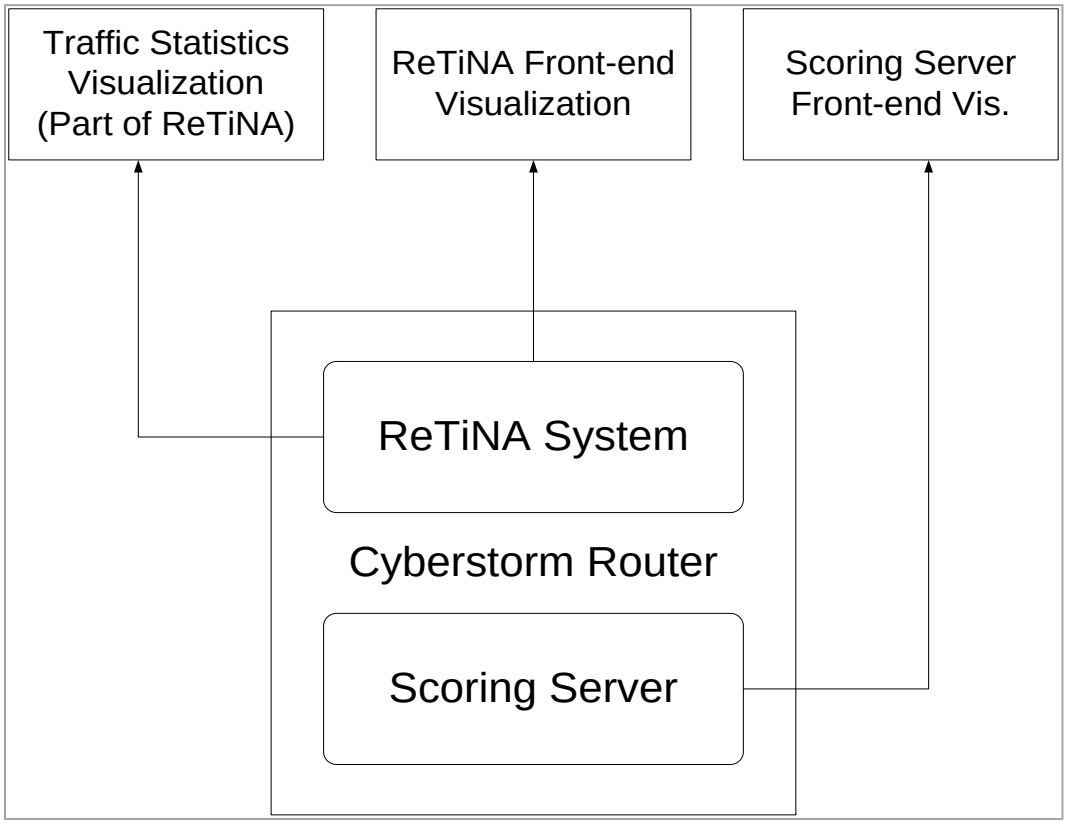

Figure 1. Cyberstorm System Data Flow Diagram

2.2 Product Functions

The ReTiNA system is comprised of five major components:

- Node Detection - Provides real time analysis of active nodes on the system
- Attack Detection - Detects attack attempts between Cyberstorm teams
- Traffic Statistics Gathering - Gathers traffic information between subnets
- ReTiNA Communication Interface - Module for transmitting information from ReTiNA modules to the front-end system for display
- Traffic Statistics Display - Displays the results of the Traffic Statistics Gathering module for spectators

These systems are presented in greater detail in the System Features (5) section of this document. Figure 2 shows a basic data flow diagram for the ReTiNA system.

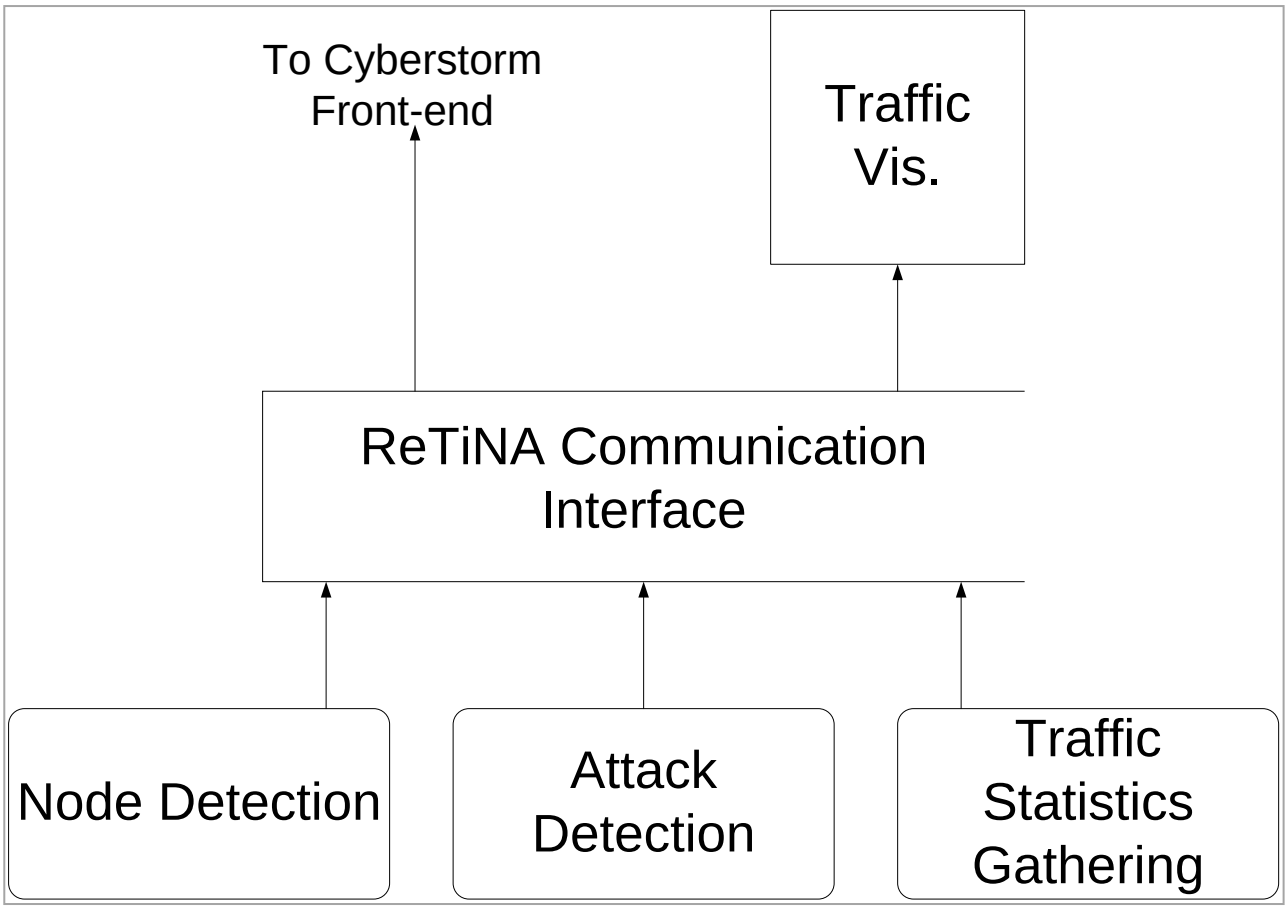

Figure 2. ReTiNA System Data Flow Diagram

2.3 User Classes and Characteristics

The primary users for the ReTiNA system are the administrators of the Cyberstorm competition or similar event. This is a user who requires a real-time analysis of network activity as a display for the spectators of the event. It is this user class the system was designed to satisfy.

A second potential user class is a system or network administrator who needs a real-time situational awareness of activity on their network. While this system is explicitly designed for this type of user, a minimal amount of configuring can be done to make the system work in this scenario.

2.4 Operating Environment

The ReTiNA system is designed to run on a Linux system running 2.6.X or higher Kernel. The hardware requirement is that the system is on a machine that can run the external software dependencies adequately. Since the ReTiNA system is designed to work as a part of the Cyberstorm system, ReTiNA must not interfere with the operation of the Cyberstorm Scoring Server or Front-end Systems. In addition, ReTiNA must not interfere with regular network traffic. Specific information about which software components must be run as a part of the ReTiNA system can be found in the Software Interfaces (3.3) section of this document.

2.5 Design and Implementation Constraints

The primary design constraints on the ReTiNA system are related to its communications requirements. The system had to be capable of communicating with the Cyberstorm front-end visualization system. This meant that all output had to be piped through a web server for remote access by the visualization system. In addition, there was a set network layout that the system was expected to be able to handle.

The implementation constraints are placed on the user of the system. It is the user's responsibility to ensure compatibility with their system (see External Interface Requirements). In addition, this system is distributed as-is as an open source product. It is the burden of the user to maintain the software.

2.6 User Documentation

All documentation for the ReTiNA system excluding this document can be found with the software. This documentation includes README files as well as instructions inside all of the configuration files that are meant to be easily changed.

2.7 Assumptions and Dependencies

This project assumes that you are running a Linux-based operating system and uses certain packages that are designed for this platform. Without these packages, the program will likely crash or display erratic behavior. These packages include Perl, Nmap, bash, python, snort, MySQL, PHP, and Apache. Further details about these packages and other dependencies can be found in the Software Interfaces section (3.3).

In addition, the ReTiNA system assumes that the user is the owner of the network, as certain components of the system, such as Nmap and promiscuous packet capture, can be viewed as hostile by certain parties.

3. External Interface Requirements

3.1 User Interfaces

The ReTiNA system is primarily designed as a back-end system to interact with a separate graphical interface within the Cyberstorm system. The only real user interface is the Traffic Statistics Display module, which has no user input. This module merely displays the traffic statistics on the network. All commands are issued either via a command line interface or by modifying the various configuration files. Details for modifying these files can be found in the files and in the documentation provided with the software.

3.2 Hardware Interfaces

ReTiNA is designed to work on an Intel x86 system, preferably one acting as a router. It should work on any computer that it is capable of meeting the software interface requirements.

3.3 Software Interfaces

The ReTiNA system is designed to work on any Linux-based operating system, but was only tested on a Linux 2.6.X kernel so compatibility with older versions is not guaranteed. It makes use of several third-party programs and libraries. These include:

- Snort 2.8.9.3
- Nmap 5.0.0
- IPTraf 3.0.0
- Apache 2.2.15
- MySQL 5.1.45
- JQuery 1.4.2
- JavaScript 1.8.1 using the Raphael library
- Perl 5.8.8 using the Lib::Tail library
- since 0.5-2.34 - a Linux tail-like function
- Python 2.6

Specifics about version numbers and required external libraries, as well as how to install and configure them, can be found in the documentation that comes with the software.

The modules for ReTiNA communicate with the front-end display through the ReTiNA Communication Interface (RCI) module. The RCI module uses a database wrapper that converts the various log files from other modules into database entries using a MySQL database. This database is then queried using CGI scripts running from an Apache web server to generate an XML document for the Cyberstorm front-end visualization to use as input.

3.4 Communications Interfaces

The ReTiNA system makes use of the ReTiNA Communications Interface (RCI) for all inter-module communication. The RCI uses an Apache web server with a MySQL back-end to send information to the requesting services. This output is in the form of an XML document created by a CGI script. A database wrapper function is used to convert the log files from the modules into the proper database entry.

None of this communication is secured or encrypted. It is up to the user to ensure security if that is a desired feature.

4. Development Cycle

In this section, our development cycle for the ReTiNA system will be explained. We used the agile software development methodology for creating this system, as it worked well with our small team size and modular construction schema. We split our development period into four weeklong iterations as well as a final testing and debugging period. A PERT chart for this project can be found in Appendix A.

Each system was developed relatively independently by one or two team members with discussions and suggestions shared with the whole team through meetings and our various project communication channels, including phone, email, and a Google group.

4.1 Iteration 1

Iteration 1 consisted primarily of research and establishing an XML format for the front end GUI team. During this period we established that Snort would be our primary method of extracting packet level attack information. This decision was based on Snort's reputation as an effective open-source intrusion detection system. In addition, a framework was established for documentation of the project using a modified Software Requirements Specification format. Finally, our team agreed upon methods for intra-team communication, using emails as our primary communication platform with a Google group for collaborative discussions.

The entire Cyberstorm team used a Task Tracker system to keep up with requirements and progress on an iteration basis. The ReTiNA iteration 1 tracker is presented in Table 1.

CS Capstone Task Tracker It. 1

Real Time Network Visualization Team

T-#	Tasks	Due Date	Distribution Info	Status	Remarks
1.00	XML Documentation	22–Mar–10	Full Team	**Complete**	XML Format for Front end team
2.00	Software Installation	24–Mar–10	Full Team	**Complete**	Installation of Snort, tcpdump, etc
3.00	Simple Log Parsing	25–Mar–10	Lockwood, King	**Working**	Pull IPs, simple attack tags from logs
4.00	XML outputting	26-Mar-10	Jackson	**Complete**	Convert Parse output to XML
5.00	Documentation	26-Mar-10	King	**Complete**	Compile SRS info, pamphlet form

Table 1. Iteration 1 Task Tracker

4.2 Iteration 2

Iteration 2 was most characterized by scope creep due to a misunderstanding between the ReTiNA team leader and the project manager about who would be handling node detection. Until this point, our focus had been on attack detection and network statistics. The addition of the node detection component split our team into a node detection project and an attack detection project. The decision was made to use a passive ARP based approach to node detection.

Our goal was to have a basic release of our system by the end of this iteration. However, the scope creep caused us to miss this goal. We did manage to establish a modular configuration approach to the node detection system to make it expandable to different network configurations and desired uses.

Table 2 shows the iteration 2 task tracker.

CS Capstone Task Tracker It. 2

Real Time Network Visualization Team

T-#	Tasks	Due Date	Distribution Info	Status	Remarks
1.00	XML Documentation, Node detection	29-Mar-10	King	**Complete**	Establish XML format, send to front-end and Sawyer
2.00	Traffic Monitor - Manipulating Snort Rules	30-Mar-10	Rollins, Lockwood	**Complete**	Learn how to add/modify Snort rules and what it picks up by default
3.00	Node detection - Input config	30-Mar-10	King	**Complete**	Complete config file and IP address input
4.00	XML Generation, Node detection	31-Mar-10	Jackson	**Complete**	Generate XML from my Node detection output
5.00	Node detection- Basic Complete	2-Apr-10	King	**Working**	1st Revision of Node detection, should have all basic functionality
6.00	Traffic Monitor - 1st Revision	2-Apr-10	Rollins, Lockwood	**Working**	1st Revision of Traffic Monitor, parse snort logs, generate needed output, basic attack types

Table 2. Iteration 2 Task Tracker

4.3 Iteration 3

Iteration 3 was primarily finishing up the carry-over from Iteration 2 and preparing a basic testable platform. This platform was deployed on a test router so we could gather performance information and plan ways to improve our base systems.

There were several key changes during this period. The first was a switch from using ARP tables to an Nmap ping sweep for node detection. This provided a more aggressive means of finding nodes, rather than relying on the more passive ARP based method. In addition, we began work on the communications interface with the attack detection to provide a real-time XML for the front-end visualization team to test against. The final change in this iteration was that we began more intense work on the network traffic monitor and were able to provide a crude implementation by the end of the iteration.

The task tracker for iteration 3 is in Table 3.

CS Capstone Task Tracker It. 3

Real Time Network Visualization Team

T-#	Tasks	Due Date	Distribution Info	Status	Remarks
1.00	Test Node detection on router	7-Apr-10	King	**Complete**	Need to get access to router to test node detection
2.00	Test ping sweeping vs. ARP cache	7-Apr-10	King	**Complete**	Check time of ping sweeping for active node detection
3.00	Stat tracking tools	7-Apr-10	Rollins	**Complete**	Establish method for stat tracking
4.00	Attack Detection XML fix	7-Apr-10	Jackson, Lockwood	**Complete**	Fix XML generator for Attack Detection to reflect the full output of the program
5.00	Debugging node Detection	9-Apr-10	King	**Complete**	Fix bugs in node detection. Correct bash errors
6.00	Research new rules for attack detection	9-Apr-10	Lockwood	**Complete**	Find ways to track new attack types
7.00	Stat tracking crude implementation	9-Apr-10	Rollins	**Complete**	Get a basic stat tracking system running
8.00	SRS update	9-Apr-10	Jackson, King	**Complete**	Bring SRS up to date

Table 3. Iteration 3 Task Tracker

4.4 Iteration 4

Iteration 4 was supposed to be a time to finish the modules and begin testing. However, there were several issues within the components and development continued during this entire period.

Most of the issues were related to problems within the ReTiNA Communication Interface. There were several problems with the XML being properly generated. In addition, the front-end team requested multiple modifications to our system output to facilitate changes in their system.

Work was done refining the Node Detection by adding caching to speed up OS detection. Also, we were able to test the system on a test router and generate logs to refine our attack detection.

We were able to do some limited testing in this phase. The results from this served primarily to highlight that there were a large number of bugs within our communications system.

Table 4 contains our task tracker for this iteration.

CS Capstone Task Tracker It. 4

Real Time Network Visualization Team

T-#	Tasks	Due Date	Distribution Info	Status	Remarks
1.00	Node Detection – Add Caching	12-Apr-10	King	**Complete**	Rough Implementation of OS Caching
2.00	Generate Log files	13-Apr-10	Extras	**Complete**	Generate snort log files for Lockwood
3.00	Installation on router	13-Apr-10	King, Lockwood, Sawyer	**Complete**	Install and configure software on new router
4.00	Prepare Production Test	14-Apr-10	All team	**Complete**	Fix bugs, implement automation, prepare Production test for review in class 4/9
5.00	Review Production Test Results	16-Apr-10	King	**Working**	Review results of Production test for completeness and accuracy
6.00	Bug fixes for next Review	19-Apr-10	All team	**Working**	Make changes and fix bugs to push a final revision
7.00	Develop new rules for Snort	16-Apr-10	Lockwood, Extras	**Working**	Using new log files, develop more rules for attack detection

Table 4. Iteration 4 Task Tracker

4.5 Final Testing

Our testing time was limited because we were still doing development on several systems during this period. Most notable was the addition of the Traffic Statistics Display module, which was originally going to be handled by the front-end team. Unfortunately, the Display module was not completed in time for the final deadline. The display will properly show total traffic statistics but will not display the real-time traffic graphs correctly.

Several problems were also found in the Attack Detection module related to its treatment of traffic between gateways. We were also requested to have the Attack Detection register remote log in attempts to several services.

Most of the holds during our testing phase of the Node Detection and Traffic Statistics Gathering modules were based on their front-ends, and the modules themselves held up well to tests.

Issue Tracker - Real Time Visualization				
Issue	**Details**	**Responsible**	**Status**	**Deadline**
StatsDB not updating	The database for converting stat logs in XMLs is not being populated correctly	Sawyer	**Resolved - 4/26**	26-Apr-10
StatsDB requires timestamp	The stats database needs a timestamp implemented so Jackson's GUI can only pull recent activity	Sawyer, Jackson	**Resolved - 4/28**	28-Apr-10
Spam squelch on attack detection	Filter traffic to prevent attack floods	Lockwood	**Resolved - 4/28**	28-Apr-10
Attacks ignore gateways	Change attack detection to ignore traffic from gateway addresses	Lockwood	**Resolved - 5/3**	5-May-10
Ftp, MySQL, http attempts	Detect attempts to remote login to these services	Lockwood	**Resolved - 5/3**	5-May-10
Stats GUI	Fix the lines on the graph	Jackson	**Unresolved TODO**	5-May-10
Stats not being read correctly	The XML being sent to the Stats GUI is not being parsed correctly.	Jackson	**Resolved - 5/3**	5-May-10

Table 5. Issue Tracker for Testing Phase

5. System Features

The ReTiNA system is composed of five main modules: Node Detection, Attack Detection, Traffic Statistics Generation, ReTiNA Communication Interface, and Traffic Statistics Display. This section will provide details on the functioning of each of these features and how they work within the ReTiNA system.

5.1 Node Detection

5.1.1 Name

The name for this module is Real Time Node Detection. It is a subset of the ReTiNA system.

5.1.2 Goal

The purpose of this module is to maintain a real-time awareness of active nodes on the network. This will allow spectators for the Cyberstorm competition as well as any system administrator monitoring their network to maintain a situational awareness of all nodes active on a network. This module also provides operating system information and limited service activity information.

5.1.3 Input

All input for this module is handled through a configuration file called nodes.conf. This file allows the user to specify which ports they wish to scan, which IP addresses should be scanned, naming conventions for the individual nodes, and node grouping. Details about how to actually manipulate this file are available in nodes.conf.

5.1.4 Output

The module outputs a plain-text log file containing information on each node it finds. Depending on how the node type was configured in the configuration file, this information can include IP address, Operating System, a user-specified group and node type, and any services which may be running on the specified ports. Although the file can be easily read by any user on its own, its format is designed to be read by the ReTiNA Communication Interface.

5.1.5 Main Scenario

The primary scenario for use of this module is in the Cyberstorm hacking competition or a similar event. However, this module can be used for any situation where an administrator desires awareness of node activity on their network.

5.1.6 Pre-condition

The only pre-condition for this module is a properly formatted configuration file. It is important to note that it is up to the user to ensure that malicious code or malformed variable names are not present in the configuration file as the module does not test for this.

5.1.7 Steps

The Node Detection module loops through the team names and associated subnets, performing a separate, identical scan for each team. The steps for a single iteration of one team are presented below.

5.1.7.1 Step 1: The module loads the team's subnet, server IP list, and node type information from the configuration file.

5.1.7.2 Step 2: The module uses Nmap to perform a ping sweep of the teams network. This generates a list of IP addresses for currently active nodes.

5.1.7.3 Step 3: The module goes through each of these IPs and performs an Operating System check and an optional port scan if the IP is present on the team's server list. A caching system is employed to ensure that an OS scan is not performed if the node was recently scanned due to time considerations. If a cached IP is no longer present in the ping sweep, it is removed from the cache.

5.1.7.4 Step 4: The results of the OS scan and port scans are then output to a log file. If this log file already exists, it is archived and a new one is created for the new data. This allows for backups of node data which can be reviewed or replayed at a later date.

5.1.8 Post-condition

The post-condition of this module is the addition of a log file and the archiving of any old log files. No other changes to the system occur as a result of this module.

5.1.9 Exceptional Scenario

An exception can occur if the configuration file is malformed or if the code is not run with proper privileges. Most errors in either of these conditions will not prevent the module from executing, but will cause anomalous output. It is up to the user to ensure that their configuration file is correct and the code is run with proper privileges, mainly for the Nmap scans.

5.2 Attack Detection

5.2.1 Name:

The name for this model is Real Time Attack Detection. It is a subset of the ReTiNA system.

5.2.2 Goal:

The purpose of this module is to maintain a real-time awareness of attacks on the network. This will allow spectators for the Cyberstorm competition to maintain situational awareness of attacks going on between the teams during the competition.

5.2.3 Input:

All input for this module is handled through the snort intrusion detection system. The module includes a modified rule set and configuration files. Once the module is installed on the system there is no required changing of these configuration files.

5.2.4 Output:

The module outputs a plain-text log file containing information on each attack detected, including a timestamp of the attack, a time for the attack to end in visualization, source IP address and team, destination IP address and team, and attack type. This log file can easily be read by any user, but it is formatted to be read by the ReTiNA Communication Interface.

5.2.5 Main Scenario:

The primary scenario for use of this module is in the Cyberstorm hacking competition or similar situation. This module will not be useful outside of this context because some of the modified rules and configuration files will not be compatible with everyday use.

5.2.6 Pre-condition:

The module assumes that snort is configured correctly on your machine, and that the configuration files and rule sets are installed correctly.

5.2.7 Steps:

The Attack Detection module starts the snort intrusion detection system and follows the alert file while parsing its contents for relevant information. Starting the module invokes the following steps:

5.2.7.1 Step1: The module starts the snort intrusion detection system.

5.2.7.2 Step 2: The module begins tailing snort's alert file, pulling the appropriate data and placing it an extraction file named "logextraction." This extraction file is kept for recording purposes and can get quite large during long running periods and will be truncated on each run of the module for this reason.

5.2.7.3 Step 3: The "logextraction" file's latest additions are then sent to "ReTiNAlog" for the ReTiNA Communication Interface to add to its database.

5.2.7.4 Step 4: The "ReTiNAlog" file is truncated after being used by the ReTiNA Communication Interface to prevent attacks from being added to the database multiple times.

5.2.8 Post-condition

The post-condition of this module is the addition of a log file named "alert.fast" in the home directory, as well as a log of the program's output in the directory it is installed named "logextraction." One more file in the program's directory, "ReTiNAlog," is created and will contain either the latest entries from "logextraction" or nothing at all.

5.2.9 Exceptional Scenario

An exception can occur if snort is not installed correctly or any of the configuration files or rules are in the wrong place. It is up to the user to make sure these are installed in the correct directories.

5.3 Traffic Statistics Gathering

5.3.1 Name

The name of this module is Traffic Statistics Gathering. It is a subset of the ReTiNA system.

5.3.2 Goal

The purpose of this module is to measure the specific traffic between the subnets of the Cyberstorm competition network. This will allow the anyone at the Cyberstorm competition to see how much traffic is being sent between all the teams specifically. This module is designed to show the data transfer between the red, blue, white team in the following fashion: red to blue, blue to red, red to white, blue to white, all the output of white, and the total data transferred between all nodes.

5.3.3 Input

The input for this module will be the from a program called IpTraf, a bandwidth monitoring program. The plain-text log file generated by this program is then parsed by my module.

5.3.4 Output

The module outputs a plain-text log file containing information on the specific number of bytes transferred between the teams as previously outlined. This is then placed into the database for long term storage and retrieval.

5.3.5 Main Scenario

The scenario for which this module was created is in the Cyberstorm hacking competition. However, this module can be easily reconfigured for use where an administrator want so monitor the traffic between subnets.

5.3.6 Pre-condition

The precondition for this module is a system with IpTraf installed (Linux based systems only at the current time). Also if different subnets besides the ones specified for the Cyberstorm competition are desired, the module will need to be changed slightly.

5.3.7 Steps

The Traffic Statistics Gathering module works in a loop parsing the output from IpTraf. It has the following steps:

5.3.7.1 Step1: The module starts IpTraf in the background which generates a log file.

5.3.7.2 Step 2: The module then parses this log file in such a way to extract the data traffic between the subnets of the Cyberstorm system and places that in a plain-text file.

5.3.7.3 Step 3: These results are stored into the database and the original log file created by IpTraf is cleared and begins to be repopulated.

5.3.7.4 Step 4: The module waits for a predetermined time of five seconds and repeats steps 2 and 3 until the module is terminated.

5.3.8 Post-condition:

The post-condition of this module is the addition of two log files and modification of the ReTiNA Communication Interface database.

5.3.9 Exceptional Scenario:

If IpTraf is not started or stopped correctly, the data traffic could be either all 0's or initially a very large number. The very large number would be generated after an extended period of time where the log file has been populated and not cleared.

5.4 ReTiNA Communication Interface

5.4.1 Name

The name for this model is ReTiNA Communications Interface. It is a subset of the ReTiNA system.

5.4.2 Goal

The primary purpose of this module was to take the information gathered by the Real Time Node Detection, Attack Detection, and Traffic Statistics Gathering Modules and relay it to the front-end visualization modules. The secondary objective was to maintain a database with the bandwidth statistics information and a list of all detected attacks from throughout the competition.

5.4.3 Input

This module receives input from the Real Time Node Detection, Attack Detection, and Traffic Statistics Gathering Modules in the form of plain-text log files. Each log file is specifically formatted for each module so that information can be easily parsed and then stored in the database.

5.4.4 Output

The module outputs generates a separate XML document for each of the three input modules: Real Time Node Detection, Attack Detection, and Traffic Statistics Gathering. Each of these XML documents contains all the information passed from each of the input modules. The XML documents are formatted specifically so that the front-end teams will be able to easily parse the data. The XML documents are passed to the front-end modules by means of printing them to a webpage (each XML document has its own unique URL) which the front-end modules can access and read.

5.4.5 Main Scenario

The primary scenario for use of this module is in the Cyberstorm hacking competition or a similar event. This module will not work in any other scenario without adjustment. The log parsing, database table, and XML document format were all designed specifically for use in the context of the Cyberstorm hacking competition.

5.4.6 Pre-condition:

There are several pre-conditions necessary in order for this module to work properly. Python, MySQL Server, and an HTTP server must be installed on the computer being used. MySQLdb, a database module for Python, must also be installed. A configuration file must be present containing the location, name, username, and password of the database being used. Also the database schema and tables must be constructed beforehand. The format for the database is as follows:

Schema: ‘mydb’ (can be any name, so long as it is specified in the configuration file)
Tables: ‘stats’, ‘attacks’, ‘nodestats’

stats:

statsid	RtB	BtR	White	RtW	BtW	assoctimestamp

statsid, assoctimestamp => type INT
RtB, BtR, White, RtW, BtW => type VARCHAR(45)

attacks:

attacksid	src	srcteam	dest	destteam	type	time	timetodie	assoctimestamp

attacksid, assoctimestamp => type INT
src, srcteam, dest, destteam, type, time, timetodie => type VARCHAR(45)

nodestats:

nodestatsid	ip	team	os	type	services

nodestatsid => type INT
ip, team, team, os, type, services => type VARCHAR(45)

5.4.7 Steps

When executed the module will parse the input log file and insert its information into the database. Also when requested, it will query the database and generated an XML document containing the information requested for by the front-end modules. The steps for the movement of information from the backend modules to the front-end modules are shown below.

5.4.7.1 Step 1: The module parses each input log.

5.4.7.2 Step 2: The module inserts the information into the appropriate table into the database.

5.4.7.3 Step 3: The module requests from the database the specific information requested by the front-end modules.

1.1.7.4 Step 4: The module creates a XML document from the information gathered from the database and prints it to the webpage to be read by the front-end module

5.4.8 Post-condition

The post-condition of this module is the entry of all information parsed from the log-files into the database and the generation of an XML document that is printed to the webpage. No other changes to the system occur as a result of this module.

5.4.9 Exceptional Scenario

An exception can occur if the configuration file is does not contain information related to a viable database. Malformed log files will also prevent the module from parsing and inserting into the database correctly.

5.5 Traffic Statistics Display

5.5.1 Name

The name for this module is Traffic Statistics Display. It is a subset of the ReTiNA system.

5.5.2 Goal

The purpose of this module is to display traffic statistics between the three teams during the Cyberstorm competition. This will allow for spectators to see real-time traffic activity by looking at a simple interface.

5.5.3 Input

This module gathers XML data from the statistics database through a CGI script. The CGI script passes the XML data of all the network activity since the last update.

5.5.4 Output

The results of the module are visuals of traffic activity. Three line graphs show the outgoing traffic of each team to the other servers over a sixty second time interval. Also, the total traffic from the Blue to Red team, from the Red to Blue, and White to both teams is displayed. These visuals are created in JavaScript using the Raphael library. The request and updates to the statistics are done using the Jquery Ajax library. It should be noted that the line graphs are not displaying properly as of the final ReTiNA release version that comes with this document.

5.5.5 Main Scenario

The primary usage of this module is for displaying traffic statistics for the Cyberstorm competition. However, if this module and the Traffic Statistics Gathering module are modified for a different network environment, the Traffic Statistics Display could show traffic between other systems.

5.5.6 Pre-condition

The pre-condition for this module is a properly formatted XML being presented either in the ReTiNA Communication Interface or at another source if the code has been modified.

5.5.7 Steps

The Traffic Statistics Display updates the graphical statistics based on a time interval (usually five seconds) by requesting a CGI script that returns an XML containing the latest activity.

5.5.7.1 Step 1: The module draws the initial statistics to the screen and gets the current system time.

5.5.7.2 Step 2: After a certain time interval, the module requests the latest statistics since the last request. The graphical statistics are updated with the latest information.

5.5.7.3 Step 3: The module continues to makes request to the database until the end of the competition or the application is terminated.

5.5.8 Post-condition

The post-condition for this module is a graphical display of byte level network traffic.

5.5.9 Exceptional Scenario

If no data is return or if there is an error when the request to the database is made, the module assumes that nothing has occur within the time interval.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

The primary performance requirement for the ReTiNA system was that it run in real time. Because of this, all systems had to be able to process their information quickly, get their information to the front-end visualization, then begin processing again. To prevent timing issues for reading and writing between the front-end and the system, we implemented a database to which all information is written. When the front-end team is ready for input, they call a CGI script which queries the database and provides and XML to be read. All further timing issues are handled within each module.

A second performance requirement was endurance. The system will need to be run for roughly seven hours during the Cyberstorm competition. This means that the system must be able to run autonomously without error for this time period.

6.2 Safety Requirements

The only potential safety issue has to do with policy rather than a software implementation. Due to the nature of some of the modules, it is recommended that the ReTiNA system only be run on a network the user maintains or has permission to use. Some components of the system, most notably Nmap, might be viewed by some administrators as an aggressive behavior.

6.3 Security Requirements

The ReTiNA system makes no attempts to ensure network security within its modules or communication structures. It is left to the user to ensure that their system is adequately protected from exploitation, including in the web server and database portions of the ReTiNA system.

6.4 Software Quality Attributes

The ReTiNA system was built with the primary attributes of robustness and correctness. The system must be able to run for the duration of the Cyberstorm competition and must accurately reflect the current traffic on the network for the spectators.

Once these two attributes where met, the system gained the additional portability, reusability, and flexibility requirements. The goal was to package the system in such a way that it could be installed on any Linux system and run with a wide variety of network configurations. The reason for this was to share the system with other institutions interested in its features as an infrastructure for hacking competitions such as Cyberstorm.

Appendix A: PERT Chart