DATABASE INTEGRATION AND GRAPHICAL USER INTERFACE FOR CYBER
DEFENSE SCORING SYSTEM

Venkata Lakkaraju
B.Tech., P.R.R.M, Jawaharlal Nehru Technological University, India, 2004

PROJECT

Submitted in partial satisfaction of
The requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2009
DATABASE INTEGRATION AND GRAPHICAL USER INTERFACE FOR CYBER
DEFENSE SCORING SYSTEM

A Project

by

Venkata Lakkaraju

Approved by:

_________________________________________________________, Committee Chair
Dr. Isaac Ghansah

_________________________________________________________, Second Reader
Prof. Richard A Smith

_________________________________________________________
Date

ii
Student: Venkata Lakkaraju

I certify that this student has met the requirements for format contained in the University format manual, and that this Project is suitable for shelving in the Library and credit is to be awarded for the Project.

Dr. Cui Zhang, Graduate Coordinator

Department of Computer Science
Abstract

of

DATABASE INTEGRATION AND GRAPHICAL USER INTERFACE FOR CYBER DEFENSE SCORING SYSTEM

by

Venkata Lakkaraju

A cyber defense competition is a competition where teams compete and learn how to defend and maintain computer security. The competition provides ground to learn the real time scenarios like defending security loopholes and adding new software or services as a typical IT company would operate. The competition requires a scoring engine which can automate the process of scoring the teams. The scoring system facilitates the judges of the competition to view the scores of each team and at the same time providing the details of how the teams performed during the competition. The main goal of this project is to build a system which is a part of scoring system, which will store the network monitoring data in a database and provide a graphical user interface which shows the results based on the data stored in the database for each team participating in the competition.

The network monitoring data collection is achieved using NDO (Nagios Database Objects) Utility, which is a plug-in for an open source network monitoring tool called Nagios. NDO Utilities configured along with Nagios acts as an interface which imports the network monitoring data into MySQL database. MySQL procedures are used to pull
the data from the database and calculate scores for each team based on the SLA (service level agreement). The graphical user interface, developed in PHP, helps judges of Cyber defense competition to answer different questions from teams participating by providing graphical reports and email scores on frequent basis. The project also implements different architectures of NDO Utilities to handle the database failures and balancing the load on different servers. The project comes with auto installation script for the software, for future use. The software was successfully used to score cyber defense competitions held at California state university, Sacramento campus.

____________________________, Committee Chair
Dr. Isaac Ghansah

Date: __________________________
I take this opportunity to thank all the people who are responsible for the successful completion of this project.

I express my sincere gratitude to Dr. Isaac Ghansah and Prof. Smith for giving me an opportunity to work under their guidance. This project helped me learn something very new, which I have not learned in my coursework.

I would like to thank all the white team members for providing their valuable suggestions for improving the GUI for the scoring system.

I thank my family for their continuous support. I would also like to thank all of my friends at CSUS for making my school days memorable.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. NDO UTILITIES</td>
<td>5</td>
</tr>
<tr>
<td>2.1 NDO Utilities Components</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Different Configurations Possible With NDO Utilities</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Pre-requisites for Installing and Proper Running of NDO Utilities</td>
<td>9</td>
</tr>
<tr>
<td>3. APPLICATION DESIGN</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Requirements Considered</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Overview of The Application</td>
<td>11</td>
</tr>
<tr>
<td>4. IMPLEMENTATION</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Installation and Configuring NDO Utilities</td>
<td>17</td>
</tr>
<tr>
<td>4.2 Installation and Configuring PHP</td>
<td>20</td>
</tr>
<tr>
<td>4.3 NDO Utilities Database Model</td>
<td>22</td>
</tr>
<tr>
<td>4.4 Calculating Scoring Based on SLA</td>
<td>25</td>
</tr>
<tr>
<td>4.5 PHP Web Pages</td>
<td>31</td>
</tr>
<tr>
<td>4.6 Create Back of Data</td>
<td>33</td>
</tr>
<tr>
<td>4.7 Implementing Distributed Architecture</td>
<td>33</td>
</tr>
<tr>
<td>4.8 Auto-Installation Scripts</td>
<td>38</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5. USER GUIDE</td>
<td>40</td>
</tr>
<tr>
<td>5.1 Login Page</td>
<td>41</td>
</tr>
<tr>
<td>5.2 Select Team Page</td>
<td>41</td>
</tr>
<tr>
<td>5.3 Score Page</td>
<td>42</td>
</tr>
<tr>
<td>5.4 Send Email Page</td>
<td>43</td>
</tr>
<tr>
<td>5.5 Team Details Page</td>
<td>44</td>
</tr>
<tr>
<td>5.6 Downtime History Page</td>
<td>46</td>
</tr>
<tr>
<td>6. CONCLUSION</td>
<td>51</td>
</tr>
<tr>
<td>Appendix A</td>
<td>53</td>
</tr>
<tr>
<td>Bibliography</td>
<td>60</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 2.1 NDOMOD Event Broker Module Overview</td>
<td>6</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2.2 NDO2DB Module Overview</td>
<td>8</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3.1 Flow of Data for Scoring System</td>
<td>14</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4.1 ER Diagram for Central or Core Tables</td>
<td>22</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 4.2 ER Diagram from Historical Data Tables</td>
<td>23</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 4.3 ER Diagram for Configuration Tables</td>
<td>24</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 4.4 Custom Table Structure for NAGIOS_CUSTOM_STATEHISTORY</td>
<td>25</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 4.5 Example Stored Procedure to Calculate Points for Each Service</td>
<td>28</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 4.6 Code Snippet from PHP Web Page</td>
<td>30</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 4.7 Code Snippet to Generate Histogram Graph in PHP Web Page</td>
<td>32</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 4.8 Distributed Data Collection Architecture</td>
<td>34</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 4.9 Scripts for Auto-Install</td>
<td>38</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 5.1 Login Page</td>
<td>40</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 5.2 Team Selection Page</td>
<td>41</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 5.3 Team Score Page</td>
<td>42</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 5.4 Email Scores Page</td>
<td>43</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 5.5 Default Format of the Email Message</td>
<td>43</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 5.6 Team Details Page</td>
<td>44</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 5.7 Host Summary page</td>
<td>45</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 5.8 Performance Graph</td>
<td>46</td>
</tr>
<tr>
<td>21.</td>
<td>Figure 5.9 Downtime History Page</td>
<td>47</td>
</tr>
</tbody>
</table>
22. Figure 5.10 Service Downtime Information .........................................................48
23. Figure 5.11 Downtime History Graph.....................................................................49
24. Figure 5.12 Date Range Parameters .....................................................................49
25. Figure 5.13 Downtime Performance Graph ............................................................50
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Table 4.1 List of Tables and Description</td>
<td>26</td>
</tr>
<tr>
<td>2.</td>
<td>Table 4.2 List of Stored Procedures</td>
<td>29</td>
</tr>
<tr>
<td>3.</td>
<td>Table 4.3 List of Methods in Histogram Class</td>
<td>31</td>
</tr>
<tr>
<td>4.</td>
<td>Table 4.4 List of PHP Pages and their Purpose</td>
<td>32</td>
</tr>
<tr>
<td>5.</td>
<td>Table 4.5 List of Scripts and their Purpose</td>
<td>39</td>
</tr>
</tbody>
</table>
Chapter 1
INTRODUCTION

Computer security is one of the main aspects of today’s enterprise level infrastructure. Every company strives to maintain high security and availability to provide uninterrupted services to their customers. Every field in today’s world is automated using computer systems for example, defense systems in military. As we talk of computer security, we are also in need of personnel who are trained in the security domain to keep track of the problems and to secure the network from the hackers and attackers. The personnel should learn to perform when there is hostile condition created by hackers or attackers who try to break the security and perform malicious activities and hence practical experience is very important for the people who work in this domain.

During 2001 United States military academy thought of doing an academic exercise which could be termed as originating point of Cyber Defense Competition [1]. Cyber defense competition is once such competition where teams compete and learn how to defend and attack systems to better understand how things work in the real time. There were several other types of competitions known as “Capture the Flag” and “Attack defend” [1] that were held at many places such as DEFCON, prior to these cyber defense competitions held by educational institutions. Scoring for the competitions such as “Capture the flag” was done based on flag associated with services. Whoever sets the flag for services would get the points. After United States military academy, a committee
consisting of educational institutions, government representatives and students [1] held a
meeting at San Antonio 2004 to discuss the standards and guidelines for Cyber defense
competitions, resulting in National collegiate competition. The National collegiate
competition conducts regional competitions and winners of regional competitions are
selected to compete in National collegiate competition [2]. In the terms set by National
collegiate competition, there would be two teams in the competition one is attackers and
the other defenders. Defenders will be assigned a group of server machines which they
have to defend, as attackers try to hack those server machines. Defenders must be capable
of securing their network as well as machines so that attackers cannot hack into their
systems. If attackers gains access to the systems of defenders they score points and in the
same line defenders loose points when they fail to maintain the security of their systems.
The defenders too score points or can balance the points they lost by working on Injects.
Injects are the tasks they have to perform by doing some installations of new software or
maintenance of their existing software. The judges give them these tasks at frequent time
intervals which students have to perform in a certain time limit. This resembles the
maintenance work in a typical IT company. Students learn how to maintain the security
by doing the maintenance at the same time. There are two rounds for the competition.
The first round would have some teams on the defending side and some on the attacking
side. In the second round attackers do the defending task and defenders vice versa.

Since there is a need for such defense competitions to get hands-on experience, there is a
similar need for a system in place which can grade such competitions. There were other
scoring systems used in competitions such as “Capture the flag” where points were given based on the flag set with each service, but the cyber defense competition is different from “Capture the flag” competitions. Many regional competitions such as Pacific Rim cyber defense competition [3] had scoring engine developed in using Python/MySQL, but software is not available for public use.

Hence project aims at providing a solution which is extension to the network monitoring tool to score the performance of the servers and services on those servers. The proposed system provides a simplified user interface which is useful in scoring cyber defense competition. The project automates the scoring for the defending machines by providing details of how good the services are running on the defender’s machines. The project integrates the network monitoring tool called Nagios [4] with an open source database integration tool known as NDO Utilities (Nagios Database Objects Utility). NDO Utilities [5] once integrated with Nagios [6] enables us to store the network monitoring status into a database (MySQL). As data is being collected into the database, the information is shown on web pages using PHP [7].

The project helps the judges of the competition view the performance of teams, email the scores to teams, answering questions from students using graphical reports. The data will be backed up every hour to provide a backup in case of a system failure. A distributed architecture is implemented to collect the data from different machines and send it to a
centralized database to balance the load. These features enable the scoring system to be
safe from database or system failures.

The report is further organized as follows:
Chapter 2 provides information about how NDO Utilities works. Chapter 3 deals with the
design part of the application and it outlines all the requirements taken into consideration.
Chapter 4 discusses the implementation of the requirements outlined in Chapter 3.
Chapter 5 provides the user guide and site map for the web pages. Chapter 6 concludes
the report. It also includes the summary of the work.
Chapter 2
NDO UTILITIES

NDO Utilities is an open source add-on for Nagios. Nagios operates based on custom configuration which has to be setup and events, such as checking a service if it is running or not. NDO Utilities stores all these events and configuration data into the database. This data can be easily retrieved for getting informative results. Every Nagios installation can be termed as an “Instance”. We can have multiple instances of Nagios sending event information to a single NDO Utilities instance or multiple instances (Which is yet to be implemented by NDO Utilities team). Nagios has all configurations about what needs to be monitored and where it has to be monitored. Each Nagios instance has to be named to uniquely identify the instance.

For the implementation of this project NDO Utilities version 1.4b7 has been used to store the data from Nagios 3.0.3 version to a MySQL database. NDO Utilities currently supports two kinds of databases to store the data one is MySQL [8] and the other is PostgreSQL [8]. Future releases might include compatibility to store data into oracle database as well. Components are being developed to store the data to multiple databases from single NDO2DB daemon. This might solve the problem of backing up the data to different storage media to avoid losing data if one the machines crash during the competition.
2.1 NDO Utilities Components

The following are the components of NDO Utilities [8]

- **NDOMOD Event Broker Module**: NDO Utilities has an event broker module as a file NDOMOD.O which sends the event data from Nagios to NDO2DB daemon. Nagios has to be installed with event broker support since NDOMOD is an event broker module which is included in the configuration of Nagios configuration files. The event broker module can output these events through three different types of output like file, Unix Domain Socket, TCP Socket. We can select any type of output depending upon our requirement. File output is useful if we want to send the file to some remote location using SSH which is more secure than Unix Domain socket and TCP Socket. If the Nagios instance is on same machine

![NDOMOD Event Broker Module Overview](figure-2.1.png)
as NDO Utilities any kind output will be secure. Figure 2.1 taken from reference [8] shows the overview of NDOMOD event broker module.

- **LOG2NDO Utility:** Whenever a log file is generated it can be processed by this utility to convert the log file into a format which NDO2DB daemon understands. The converted format is then sent to the NDO2DB daemon using either Unix Socket Domain or TCP Socket (This would the output type NDO2DB daemon accepts information). This utility comes into play when we want to store the historical information from Nagios into database.

- **FILE2SOCK Utility:** This utility takes a file as an input and sends it to either UNIX or TCP socket. The input file can be a file coming from LOG2NDO utility or standard output file from the NDOMOD event broker module (If the output type is set to standard file). Once the NDOMOD event broker module completes writing the data into a file it can be processed by this utility to send it to NDO2DB socket. This utility is useful if the Nagios instance and NDO2DB daemon reside in two different machines. In this case the Nagios instance generates the file in one machine that needs to be sent to database which is installed on a different remote machine.
**Figure 2.2 NDO2DB Module Overview**

- **NDO2DB Utility**: NDO2DB is the important component of NDO Utilities. This utility runs as daemon in the background to collect the information sent through the TCP or Unix Socket from the NDOMOD event broker module. This daemon also collects the data sent by LOG2NDO module to store the log information from the Nagios instance. Whenever started this daemon runs as a standalone process listening to the TCP or Unix Domain Socket. When multiple clients try to send the data from different machines the daemon spawns in to multiple instances to handle them. Hence multiple NDOMOD broker modules can send data to a single NDO2DB daemon. Figure 2.2 taken from reference [8] shows the NDO2DB module overview.

**2.2 Different Configurations Possible with NDO Utilities**

- **Single Nagios Instance**: This configuration includes a single Nagios instance storing data into single database.
• **Multiple Standalone Nagios Installations:** This configuration has multiple instances of Nagios sending information to a single database. It is important in this kind of installation to have unique names for each instance of Nagios to avoid duplicate names or confusion.

• **Multiple Nagios Installations in a Fail-over, Redundant or Distributed Environment:** In this configuration different Nagios installations share the load between them to balance the performance. Another scenario can have multiple Nagios instances where only one of them is active (Performing checks to see if services are running) and others stand-by to provide backup when the primary instance crashes. Redundant environment have multiple Nagios doing the same work as the other to have backup if one of the instances fail.

### 2.3 Pre-requisites for Installing and Proper Running of NDO Utilities

- Any Linux operating system
- Nagios 3.0 or later installed to collect the data with event broker support
- MySQL 5.0 database
- If we are using TCP socket output, NDO Utilities uses port 5668 as default port which needs to be open for the daemon to receive event data.
- mysql-devel and mysql-client packages for NDO2DB module to connect to database
Chapter 3
APPLICATION DESIGN

This chapter provides an overview of requirements and design of the application. It also includes description of components of the application.

3.1 Requirements Considered
The following are the requirements that are considered for developing the application based on what functions the application has to perform

- Integrate Nagios with a database to make the data monitored by Nagios available in a database
- Design a Real time GUI which has following capabilities
  - Judges of the competition can view scores at anytime
  - Check different services available services on each machines assigned to teams
  - Check performance of each host and its services by showing performance graphs
  - Email Scores to the teams
  - Be able to answer questions from Students regarding scoring
- Create Database scripts to calculate points based on SLA defined for the Competition
• Be able to gather data from multiple servers and send them to centralized Database
• Provide an auto installation script for future deployments
• Implement the system and test in real time environment

3.2 Overview of the Application

The application developed in this project aims at providing an extension to NDO Utilities to fetch the data from MySQL database and display the information in a simple web interface using PHP pages. Figure 3.1 shows the flow of data in the application. The following are different components used in this application

• Nagios
• NDO Utilities
• MySQL DB
• PHP Web Pages

Nagios: Nagios forms the dependency for this application. All the defenders machines which should be monitored are configured in Nagios. All the events occurring in these machines are then recorded by Nagios and sent by NDO Utilities Event Broker Module. The following steps are implemented in Nagios

  ▪ Determine machines and host addresses of those machines which have to be monitored
- Identify services running on these machines
- Compile Nagios with event broker support

**NDO Utilities:** This component integrates Nagios with the MySQL database. The configuration includes selecting the type of output which Event Broker Module outputs the data for e.g. TCP Socket, Unix Domain Socket or File output. Once everything is configured NDO2DB daemon is started to collect the data from Event broker and store it to the database. All the setups that have to be completed to achieve this will be explained in Chapter 4 where we discuss the implementation of the requirements. This component includes configuring the NDOMOD event broker module and NDO2DB daemon. Once Nagios is started the NDOMOD loads along with it to collect all the events and sends it to NDO2DB daemon running using the Unix Domain Socket. The following components are pre-requisites for this component.

- Nagios 3.0.3
- MySQL database
- Mysql-client and mysql-devel packages ( NDO utilities use header files from these packages to communicate with MySQL database)

**MySQL Database:** The application uses MySQL database to store all the data. A new database called NAGIOS will be created to store all the data coming from NDO event broker module. When the NDO Utilities is compiled it has to done with MySQL support,
since we are using MySQL as backend database. All the tables which are used to store the data will be created using the script which comes along with the NDO Utilities package. As data is stored in to the tables, data from these tables can be pulled and shown on the web using PHP pages. The NAGIOS database has two different kinds of tables, configuration tables and dynamic tables. Configuration tables store the configuration data of Nagios instance and the dynamic tables store the information about the events accruing the Nagios. This component acts as storage for the application to store all the data. Different procedures are used to calculate the points lost by the defending team. These procedures are stored packages in MySQL database which can be called from a query in PHP pages after connecting to database. The competition rules also include an SLA based on which the points are deducted. All the custom procedures written for the purpose of the scoring come under custom objects in NAGIOS database. We also have some tables which stores temporary table for certain period of time to show it on the web.

The following are some of the pre-requisites for this component

- Access to create database in MySQL
- Access to create, update, delete on tables in NAGIOS database
**Figure 3.1 Flow of Data for Scoring System**

**PHP Web Pages**: This component forms the graphical user interface for the scoring system. Once the data is stored in the database, judges can access the data by authenticating themselves using these web pages. The web pages enable the judges to see the performance of different teams at any point of time. They can also see points lost by the defender’s team and send them scores by email. The judges can also see which machines belong to which team and what are the services which are hosted on them.
Performance graphs are available for each server based on time versus % up time of each of the services running the machine. These web pages are available on the machines assigned to the judges of the competition. If public IP are assigned to these systems judges can access these web pages remotely. This enable judges to see the teams performance from anywhere instead of going to team’s workstations and check out their status.

The following are some of the pre-requisites which are required for this component

- Access to NAGIOS database
- Php
- Php-gd package (This package is needed to display the histogram graphs for showing the service downtime of a server)
- Apache [9] is web server to host the web pages.
- Sendmail [10] has to be configured on the hosting Linux server for sending scores from the webpage
Chapter 4

IMPLEMENTATION

This chapter provides insight on how the requirements are implemented to develop the application. This chapter also describes the setups needed prior to development of the application. All the commands provided in this chapter are specific to CentOS 5.X [11] operating system. The application is developed specifically for this operating system. Following points are the sequence of steps followed during the implementation

- Install and configure NDO Utilities
- Install and configure PHP
- Create Nagios Database and run NDO Utilities database scripts
- Understand NDO Utilities database model and figure out tables which store the information about events occurring in Nagios
- Create database functions to calculate points based on SLA
- Create database functions to calculate service performance of each service hosted
- Create function and a temporary table to store the downtime history for each service
- Design web pages using PHP
- Create hourly backup of data
- Implement distributed architecture for NDO Utilities, collect data from different systems and send it to a centralized database
- Create auto-installation script for future use.
4.1 Installation and Configuring NDO Utilities

NDO Utilities version 1.4b7 is used for developing this application. This version of NDO Utilities is compatible with Nagios 2.x onwards. Nagios version 3.0.3 is setup for this application to monitor the network. The latest version of NDO Utilities can be downloaded from [http://www.nagios.org/download](http://www.nagios.org/download). The following are steps for installation once the tar ball is downloaded from the NDO Utilities website as in reference [8].

- Copy the downloaded software to ~/downloads folder
- Untar the file using the command
  ```bash
tar xzf ndoutils-1.4b7.tar.gz
  ```
- Change to directory created under downloads after untar
  ```bash
  cd ndoutils-1.4b7
  ```
- Use make command to compile the build
  ```bash
  ./configure --with-mysql-lib=/usr/lib/mysql
  make
  ```
- Once the out files are created after the make copy the following files to /usr/local/nagios/bin folder ndomod-3x, ndo2db-3x, file2sock, log2ndo using the following command. This ends the installation for the NDO Utilities.
  ```bash
  cp ndomod-3x ndo2db-3x file2sock log2ndo /usr/local/nagios/bin
  ```
Now we have to configure NDO Utilities to suit our needs we get two files with NDO Utilities which have the configuration information, they are ndomod.cfg and ndo2db.cfg. ndomod.cfg has the configuration setups for the NDO event broker modules and ndo2db.cfg has the configuration setups needed for the NDO2DB daemon. Before beginning the configuring NDO Utilities make sure you have the following setting in /usr/local/nagios/etc/nagios.cfg

```bash
event_broker_options = -1
```

Following steps are used for creating NAGIOS database in MySQL

```sql
CREATE database nagios;
GRANT ALL PRIVILEGES ON *.* TO 'nagiosdb'@'localhost' IDENTIFIED BY 'nagiosdb';
```

Once database is created the database scripts that came along with NDO Utilities to create tables is run using the following command after switching to folder db

```bash
./installdb -u nagiosdb -p nagiosdb -h localhost -d nagios
```

Now that the database is ready we have information to configure ndomod.cfg and following are the options which were setup

```plaintext
instance_name=default
//# OUTPUT TYPE
output_type=unixsocket  # Unix Domain Socket output is selected
 # for this application

//# OUTPUT
output=/usr/local/nagios/var/ndo.sock
 # TCP PORT
tcp_port=5668
 # OUTPUT BUFFER
output_buffer_items=5000
 # BUFFER FILE
```
buffer_file=/usr/local/nagios/var/ndomod.tmp
# FILE ROTATION INTERVAL
file_rotation_interval=14400
# FILE ROTATION TIMEOUT
file_rotation_timeout=60
# RECONNECT INTERVAL
reconnect_interval=15
# RECONNECT WARNING INTERVAL
reconnect_warning_interval=15
# DATA PROCESSING OPTION
data_processing_options=-1
# CONFIG OUTPUT OPTION
config_output_options=2

- Following configuration was used for NDO2DB daemon in the file ndo2db.cfg

ndo2db_user=nagios
ndo2db_group=nagios
# SOCKET TYPE
socket_type=unix
# SOCKET NAME
socket_name=/usr/local/nagios/var/ndo.sock
# TCP PORT
tcp_port=5668
# DATABASE SERVER TYPE
db_servertype=mysql
# DATABASE HOST
db_host=localhost
# DATABASE PORT
db_port=3306
# DATABASE NAME
db_name=nagios
# DATABASE TABLE PREFIX
db_prefix=nagios_
# DATABASE USERNAME/PASSWORD
db_user=nagiosdb
db_pass=nagios
## TABLE TRIMMING OPTIONS
max_timedevents_age=1440
max_systemcommands_age=10080
max_servicechecks_age=10080
max_hostchecks_age=10080
max_eventhandlers_age=44640
# DEBUG LEVEL
download_level=0
download_verbosity=1
# DEBUG FILE
download_file=@localstatedir@/ndo2db.debug
# MAX DEBUG FILE SIZE
max_download_file_size=1000000

- Now setup and configuration is done, we should start the NDO2DB daemon using the following command

  /usr/local/nagios/bin/ndo2db-3x –c /use/local/nagios/etc/ndo2db.cfg

- We can now see the data being stored in database. We can also check the logs whether everything is looking fine.

### 4.2 Installation and Configuring PHP

All the web pages used to design the user interface in this application as designed using PHP. PHP version 5.x is used to develop these pages. All the performance graphs in the application are designed using histogram class which can be downloaded as open source from following link [http://www.phpclasses.org/](http://www.phpclasses.org/). PHP is installed and all the required setups are done in Apache configuration files. Following are the sequence of steps followed to configure and install PHP [7]

- Download and install command for PHP

  yum –y install php
- Download and install command for PHP-GD package which is required for displaying the histogram graphs.
  
  `yum -y install php-gd`

- Download the histogram class from [http://www.phpclasses.org/](http://www.phpclasses.org/) and copy the php files into the folder `/var/www/html`. In order to use methods from histogram class we have to include this folder to php.ini file which is initialization file for PHP. This file is available in `/etc` folder and following is the directive to include in the file

  `include_path = ../php/includes:/var/www/html`

- The folder has to be added to `httpd.conf` file which is the configuration file for Apache http server. The configuration file includes the directory and authentication details of the web pages for the application. The `httpd.conf` file is available under the folder `/etc/httpd/conf` folder and following is the directive to include the directory in the apache configuration file.

  ```
  Require valid-user
  </Directory>

  <Directory "/var/www/html">
  Options None
  AllowOverride None
  Order allow,deny
  Allow from all
  AuthName "Nagios Access"
  AuthType Basic
  AuthUserFile /usr/local/nagios/etc/htpasswd.users
  Require valid-user
  </Directory>
  ```

This directive uses file `htpasswd.users` which is has the username and password used for Nagios.
• When all these steps are configured apache has to be re-started using the following command

   service httpd restart

4.3 NDO Utilities Database Model

It is very important to get accustomed tables created by NDO utilities database scripts as it consists of all the information which is used to display in the web pages. The tables in NDO utilities database model [13] can be divided into following categories, although we have many tables which form the database for NDO Utilities, tables which contain information used in this application are shown below in these categories.

• **Central Tables:** These tables form the core for all other tables in NDO utilities all other tables refer to either of these core tables. Figure 4.1 shows the two tables of this category.

![Figure 4.1 ER Diagram for Central or Core Tables](image)

• **Debugging Tables:** These tables are used for storing debugging information. There is only one table in this category which we are not using in this application.
**Historical Data Tables:** These tables have the historical information which occurred in Nagios, these tables might also have information about the services which are still active, but time wise these events have taken place some time back. Hence this information is important for us to calculate scoring. ER diagram in Figure 4.2 provides information about historical data tables used in this application.

![ER Diagram from Historical Data Tables](image)

**Current Status Tables:** These tables are used to store the information about the current status of the host and services being monitored once NDO2DB daemon starts running, these tables are cleared once daemon is re-started or stopped. These tables are not used in this application.
- **Configuration Tables:** These tables hold the configuration information of Nagios such as which hosts are being monitored and the services which they are hosting. Figure 4.3 shows the ER diagram for configuration tables.

![ER Diagram for Configuration Tables](image)

**Figure 4.3 ER diagram for configuration tables**

- **Custom Tables:** These tables are designed for the application which is not part of the NDO Utilities database model. This is table is used in the state history query
The table is a temporary table which stores the data using a procedure and is cleared once the data is displayed on the web page. Figure 4.4 shows the table name and the columns used in the history table.

<table>
<thead>
<tr>
<th>NAGIOS_CUSTOM_STATEHISTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_id</td>
</tr>
<tr>
<td>instance_id</td>
</tr>
<tr>
<td>object_id</td>
</tr>
<tr>
<td>start_time</td>
</tr>
<tr>
<td>end_time</td>
</tr>
<tr>
<td>total_time</td>
</tr>
</tbody>
</table>

**Figure 4.4 Custom Table Structure for NAGIOS_CUSTOM_STATEHISTORY**

### 4.4 Calculating Scoring Based on SLA

Now that we have NDO Utilities configured and table information. The data is being collected into tables the next step to implement is creating stored procedures in MySQL to calculate the performance of the services and calculate points. The SLA for the competition will be defined based on which the points will be deducted for the defenders. The points are assigned on a negative scale. The points shown in the web page are the points which have to be deducted from defenders team.

Table 4.1 describes the table names, their description and purpose used in procedures
<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>DESCRIPTION</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCES</td>
<td>Contains information about Nagios instances</td>
<td>Used to display the Nagios instance information in case of multiple Nagios instances are involved (Used for distributed architecture)</td>
</tr>
<tr>
<td>OBJECTS</td>
<td>Objects can be hosts, services, service groups or host groups</td>
<td>Contains the names of the hosts, services</td>
</tr>
<tr>
<td>HOSTS</td>
<td>Contains details of hosts like IPaddress etc</td>
<td>Used to display the IPaddresses and machine names of the Hosts</td>
</tr>
<tr>
<td>HOST GROUPS</td>
<td>Contains details of host groups</td>
<td>Used to group hosts which come under each team</td>
</tr>
<tr>
<td>HOSTGROUP_MEMBER S</td>
<td>Contains which hosts belong to the group</td>
<td>Used to relate host groups and host group members</td>
</tr>
<tr>
<td>SERVICES</td>
<td>Contains details of the services configured for each host</td>
<td>Contains details of services running on defender team’s machines</td>
</tr>
<tr>
<td>SERVICES GROUP</td>
<td>Contains services group information</td>
<td>Contains services group information</td>
</tr>
<tr>
<td>SERVICE GROUP_MEMEBERS</td>
<td>Detailed information of services group</td>
<td>Detailed information of services group</td>
</tr>
<tr>
<td>SERVICE CHECKS</td>
<td>Historical data of service check performed</td>
<td>Used to calculate down time for each service</td>
</tr>
<tr>
<td>HOST CHECKS</td>
<td>Historical data of host check performed</td>
<td>Used to calculate down time for each host</td>
</tr>
<tr>
<td>NAGIOS Custom STATE HISTORY</td>
<td>Downtime information for a service of a host in a team</td>
<td>Shows the frequencies at which the service was down during the competition</td>
</tr>
</tbody>
</table>

Table 4.1 List of Tables and Description
The following are the SLA rules used for the application developed

- Service down for over an hour: deduct 10 points
- Service down for over 2 hours: deduct 20 points
- Service down for over 3 hours: deduct 40 points
- Service down there after 40 points will be deducted for each additional hour of time

The following paragraph explains the calculation of score for a service based on example along with the code. Figure 4.5 shows the code for calculating points based on SLA. The code takes the parameters instance_id from NAGIOS_INSTANCES table, host_object_id from NAGIOS_HOSTS and service_object_id from NAGIOS_SERVICES and returns the points lost due to down time for that service. Instance_id, host_object_id, service_object_id uniquely identifies service. The down time is calculated using a query from NAGIOS_SERVICECHECKS taking column values state and end_time. The NAGIOS_SERVICECHECKS has the state values as 0, 1, 2, 3 where 0 means OK, 1 means WARNING, 2 means CRITICAL and 3 means UNKNOWN state of service. The procedure counts only records which have state not equal to 3 since state 3 specifies that Nagios is unable to find the service and is not a downtime state. Once we have records with state as 0 or 1 we can find the time difference between the end_time to get the actual time the service has been down. The end_time time stamp is in seconds and hence has to be converted to minutes and then check for the score to be deducted from the defending team’s points.
```sql
DELIMITER $$
DROP FUNCTION IF EXISTS Calculatepoints$$
CREATE FUNCTION Calculatepoints(p_inst_id INT, p_host_obj_id INT, p_serv_obj_id INT) RETURNS INT
DETERMINISTIC
BEGIN
DECLARE STATUS_FLAG INT DEFAULT 0;
DECLARE v_time INT DEFAULT 0;
DECLARE v_state INT DEFAULT 0;
DECLARE v_end_time DATETIME;
DECLARE LOOP_FLAG INT DEFAULT 0;
DECLARE PREV_TIME_STAMP DATETIME;
DECLARE PREV_STATE INT DEFAULT 10;
DECLARE POINTS INT DEFAULT 0;
-- CURSOR FOR SERVICE CHECKS
DECLARE cur_services CURSOR FOR
SELECT nsc.state, nsc.start_time
FROM nagios_servicechecks nsc
WHERE nsc.service_object_id = p_serv_obj_id AND
  nsc.instance_id = p_inst_id AND
  nsc.state < 3 AND
  nsc.start_time BETWEEN '2008-11-11 00:00:00' AND '2008-11-18 12:00:00'
ORDER BY nsc.service_object_id, nsc.end_time ASC;
DECLARE CONTINUE HANDLER FOR NOT FOUND
  SET LOOP_FLAG = 1;
-- CHECK FOR NAGIOS SERVICES
OPEN cur_services;
FETCH cur_services INTO PREV_STATE, PREV_TIME_STAMP;
SET v_time = 0;
WHILE LOOP_FLAG = 0 DO
  FETCH cur_services INTO v_state, v_end_time;
  IF v_state = 2 THEN
    REPEAT
      SET v_time = v_time + TIME_TO_SEC(TIMEDIFF(v_end_time, PREV_TIME_STAMP)) / 60;
      SET PREV_STATE = v_state;
      SET PREV_TIME_STAMP = v_end_time;
      FETCH cur_services INTO v_state, v_end_time;
    UNTIL v_state IN (0, 1) OR LOOP_FLAG = 1
    END REPEAT;
  END IF;
  SET PREV_STATE = v_state;
  SET PREV_TIME_STAMP = v_end_time;
END WHILE;
IF v_time > 60 AND v_time <= 120 THEN
  SET POINTS = POINTS + 10;
ELSEIF v_time > 120 AND v_time <= 180 THEN
  SET POINTS = POINTS + 20;
ELSEIF v_time > 180 THEN
  SET POINTS = POINTS + 70;
  SET v_time = v_time - 180;
  SET POINTS = POINTS + (v_time / 60) * 40;
END IF;
CLOSE cur_services;
RETURN POINTS;
END$$
```

Figure 4.5 Example Stored Procedure to Calculate Points for Each Service
Table 4.2 lists the name, purpose and parameters of all the functions created for to be used in PHP pages.

<table>
<thead>
<tr>
<th>NAME OF PROCEDURE</th>
<th>PARAMETERS</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculatepoints</td>
<td>Instance_id, host_object_id, service_object_id</td>
<td>Calculates points based on SLA for Service checks</td>
</tr>
<tr>
<td>Calculatehost</td>
<td>Instance_id, host_object_id</td>
<td>Calculates points based on SLA for host checks</td>
</tr>
<tr>
<td>Calculateuptime</td>
<td>Instance_id, host_object_id, date</td>
<td>Calculate performance for host, used in the histogram graphs</td>
</tr>
<tr>
<td>CalculateServPerf</td>
<td>Instance_id, host_object_id, service_id, date</td>
<td>Calculate performance for services used in the histogram graphs.</td>
</tr>
<tr>
<td>StateHistory</td>
<td>Instance_id, object_id, run_id (unique key generated for each run along with timestamp)</td>
<td>This function is used to fetch state history data in to a temporary table for display</td>
</tr>
</tbody>
</table>

**Table 4.2 List of Stored Procedures**

At the end of the first round the procedures Calculatepoints, Calculatehost and StateHistory has to be compiled with the data range for which the second round is scheduled. These procedures have the hard coded values of data range for which the
round is scheduled. These procedures are used along with the queries in the PHP pages to display the information. Next section describes how PHP pages are designed.

```php
mysql_select_db("nagios","connection");

$results = mysql_query("select ni.instance_name,
                        nhg.alias team_name,
                        SUM(Calculatehost(nh.instance_id,nh.host_object_id)) total_points
                        from nagios_instances ni,
                        nagios_hosts nh,
                        nagios_hostgroups nhg,
                        nagios_hostgroup_members nhgm
                        where ni.instance_id = nh.instance_id and
                        nhg.instance_id = ni.instance_id and
                        nhg.hostgroup_id = nhgm.hostgroup_id and
                        nh.host_object_id=nhgm.host_object_id and
                        upper(nh.display_name) <> upper('localhost') and
                        upper(nhg.alias) = upper("$_POST['Teams']")
                        group by ni.instance_name,
                        nhg.alias
                        order by nhg.alias asc");

while($row = mysql_fetch_array($result))
{ echo "<tr>
  echo "<td>".$row['instance_name']."</td>
  echo "<td>".$row['team_name']."</td>
  $calc_points= "select sum(calculatepoints(ns.instance_id,ns.host_object_id,ns.service_object_id)) points
                        from nagios_services ns,
                        nagios_hostgroups nhg,
                        nagios_hostgroup_members nhgm
                        where nhg.instance_id = ns.instance_id and
                        nhg.hostgroup_id = nhg.hostgroup_id and
                        nhg.hostgroup_id = nhgm.hostgroup_id and
                        nhgm.host_object_id = ns.host_object_id and
                        upper(nhg.alias) = upper("$_POST['Teams']")
                        group by ns.instance_name,
                        nhg.alias
                        order by nhg.alias asc");

  $row1 = mysql_query($calc_points);
  $result1 = mysql_fetch_array($row1);
  $points = $result1['points'];
  $points = $points + $row['total_points'];
  echo "<td>".$points."</td>";
  echo "<td><a href="http://localhost/ViewDetails.php?teamname=".$row['team_name'].">Click Here</a> / </td>
  echo "<td><a href="http://localhost/SendMail.php?teamname=".$row['team_name'].">Email Scores</a>;</n  echo "</td>";
no mysql_close($connection);
?
```

Figure 4.6 Code Snippet from PHP web page
4.5 PHP Web Pages

Now that the database functions are ready, we can now create PHP web pages using these database functions and tables to display the information required. Figure 4.6 shows the code snippet from one of the PHP pages in the application. It shows how the score is calculated using the native functions for mysql in php such as mysql_connect, mysql_query, mysql_fetch_array and mysql_close. The example in Figure 4.6 shows how the data is fetched from the database using these functions and displayed on the web page. The php-gd package is installed to display histograms used in some of the pages to show the % Up time for services and host status. The PHP pages use class.graph.php and class.graph.histogram.php which have the required functions to display the histogram functions. Table 4.3 describes different methods available in class.graph.histogram.php. Figure 4.7 is the code snippet used for graphs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Input Parameters</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histogram( )</td>
<td>Two Dimensional array containing data to be displayed</td>
<td>Constructor class to initialize histogram class</td>
</tr>
<tr>
<td>Draw( )</td>
<td>None</td>
<td>Method to draw the graph</td>
</tr>
</tbody>
</table>

Table 4.3 List of methods in Histogram class
Figure 4.7 Code Snippet to Generate Histogram Graph in PHP Web Page

Table 4.4 shows names of php pages developed in the application and their purpose.

<table>
<thead>
<tr>
<th>Page Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCDC.php</td>
<td>Log in page to select teams</td>
</tr>
<tr>
<td>Viewscores.php</td>
<td>To show scores of selected team</td>
</tr>
<tr>
<td>ViewDetails.php</td>
<td>To show details of the team selected</td>
</tr>
<tr>
<td>SendEmail.php</td>
<td>To send the scores to the team</td>
</tr>
<tr>
<td>DispGraph.php</td>
<td>Summary page which shows performance of each server in the defending team</td>
</tr>
<tr>
<td>StateHistory.php</td>
<td>To check the downtime history of a service and answer questions from students</td>
</tr>
</tbody>
</table>

Table 4.4 List of PHP pages and their purpose

Further information about the pages and their functions will be explained in the user-guide Chapter- 5.
4.6 Create Back of Data

It is important for the scoring system not to loose data in case of a database crash. The data needs to be backed up every hour either in local hard drive, any remote machine or pluggable drive which can store the data so that data is available after the crash. In order to achieve this cron job is setup to mysqldump for every hour the following directive would take the back up of data every hour and store in a text file. If the file has to be sent to a remote machine ssh has to be used to securely login and send the backup file. The cron job can be setup under /etc/crontab file.

01 * * * *  mysqldump –h localhost  -u nagiosdb –p nagiosdb nagios >> /backup.txt

4.7 Implementing Distributed Architecture

This sub-topic explains how distributed collection of data is implemented using NDO Utilities. The NDO Utilities is configured in such a way that the data is collected on different servers and sent to a centralized database. The figure 4.8 shows how the basic architecture works for distributed collection of data [8].
Figure 4.8 Distributed Data Collection Architecture

NDO Utilities provides two types of data collection. One of them is real time data collection and the other is Delayed gathering using FILE2SOCK utility. Real time data collection tries to collect the data as and when event occurs in Nagios instance. Delayed gathering writes Nagios events in to a standard file output which is later on sent to the socket of NDO2DB using the utility FILE2SOCK. This can be done by providing an ssh
connection to the server hosting NDO2DB daemon. In this application real time data gathering is used as there has to be no delay while collecting the data as the scores sent to the teams are accurate and up-to-date.

As Figure 4.8 depicts multiple servers are configured to use TCP socket to send data using port 5668 to the NDO2DB running on same or different server. The Nagios instance which collects the data is configured to have only the NDOMOD so that event information can be collected and output is sent to TCP socket. The centralized server is configured with a MySQL database and NDO2DB daemon running on TCP port 5668 which accepts connections from multiple servers. Once the Nagios instances on remote servers are run, remote server starts collecting the data from Nagios instances.

This architecture is useful when we want to distribute the work load to multiple servers. When there are too many servers and network devices monitored this architecture distributes the work among multiple Nagios instances. Also we can isolate the database to a different server, which explicitly maintains the database. The obvious disadvantage of this design is that we have increased network traffic. The configuration in NDOMOD.cfg should be as follows for the remote servers having NDOMOD for Nagios instance 1 in Figure 4.8

```bash
instance_name=instance1
# OUTPUT TYPE
output_type=tcpsocket  # Unix Domain Socket output is selected
# for this application
# OUTPUT
```
output=192.168.1.4
# TCP PORT
tcp_port=5668
# OUTPUT BUFFER
output_buffer_items=5000
# BUFFER FILE
buffer_file=/usr/local/nagios/var/ndomod.tmp
# FILE ROTATION INTERVAL
file_rotation_interval=14400
# FILE ROTATION TIMEOUT
file_rotation_timeout=60
# RECONNECT INTERVAL
reconnect_interval=15
# RECONNECT WARNING INTERVAL
reconnect_warning_interval=15
# DATA PROCESSING OPTION
data_processing_options=-1
# CONFIG OUTPUT OPTION
config_output_options=2

For instance 2 in figure 4.8

instance_name=instance1
# OUTPUT TYPE
output_type=tcpsocket # Unix Domain Socket output is selected # for this application

# OUTPUT
output=192.168.1.4
# TCP PORT
tcp_port=5668
# OUTPUT BUFFER
output_buffer_items=5000
# BUFFER FILE
buffer_file=/usr/local/nagios/var/ndomod.tmp
# FILE ROTATION INTERVAL
file_rotation_interval=14400
# FILE ROTATION TIMEOUT
file_rotation_timeout=60
# RECONNECT INTERVAL
reconnect_interval=15
# RECONNECT WARNING INTERVAL
reconnect_warning_interval=15
# DATA PROCESSING OPTION
data_processing_options=-1
# CONFIG OUTPUT OPTION
cfg_output_options=2

and ndo2db.cfg for machine labeled 192.168.1.4 should be

ndo2db_user=nagios
ndo2db_group=nagios
# SOCKET TYPE
socket_type=tcp
# TCP PORT
tcp_port=5668
# DATABASE SERVER TYPE
db_servertype=mysql
# DATABASE HOST
db_host=localhost
# DATABASE PORT
db_port=3306
# DATABASE NAME
db_name=nagios
# DATABASE TABLE PREFIX
db_prefix=nagios_
# DATABASE USERNAME/PASSWORD
db_user=nagiosdb
db_pass=nagios
## TABLE TRIMMING OPTIONS
max_timedevents_age=1440
max_systemcommands_age=10080
max_servicechecks_age=10080
max_hostchecks_age=10080
max_eventhandlers_age=44640
# DEBUG LEVEL
dbg_level=0
dbg_verbosity=1
# DEBUG FILE
dbg_file=@localstatedir@/ndo2db.debug
# MAX DEBUG FILE SIZE
max_dbg_file_size=1000000
4.8 Auto-Installation Scripts

One of the major requirements of this application is to have an auto installation script which would automate all the installation steps in this application. This would enable project to be a ready to configure and install application. Since the base operating system for Nagios is Linux a shell script would serve the purpose of installing the software. The auto installation script created for this application is a shell script which executes all the installation steps. Figure 4.9 shows different scripts which are involved in the auto installation. Table 4.5 lists scripts and their purpose.

![Figure 4.9 Scripts for Auto-Install](image)

The whole scripts would be available in the folder Software, this folder has to be copied to any home folder and then using the following command the main script has to be executed. This script is based on another installation script for Nagios [4].

```
./InstallConfigureDB_GUI.sh
```
<table>
<thead>
<tr>
<th><strong>Script Name</strong></th>
<th><strong>Purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>InstallConfigureDB_GUI.sh</td>
<td>Main installation script which calls other scripts</td>
</tr>
<tr>
<td>http_config</td>
<td>Contains configuration information for httpd.cof</td>
</tr>
<tr>
<td>statehistory.sql</td>
<td>Creates custom table for owntime history and function to retrieve information</td>
</tr>
<tr>
<td>DBscript.sql</td>
<td>Creates Nagios database and user to access the database</td>
</tr>
<tr>
<td>Calculatepoints.sql</td>
<td>Code for Function used in webpages</td>
</tr>
<tr>
<td>CalculateServPerf.sql</td>
<td>Code for Function used in webpages</td>
</tr>
<tr>
<td>Calculateuptime.sql</td>
<td>Code for Function used in webpages</td>
</tr>
<tr>
<td>Calculatehost.sql</td>
<td>Code for Function used in webpages</td>
</tr>
<tr>
<td>Webpages</td>
<td>This folder contains all the webpages used by the application</td>
</tr>
</tbody>
</table>

**Table 4.5 List of Scripts and Their Purpose**

When all the commands are executed the script generates a log for the while installation giving you information about what packages are updated or installed on the system. The file of the log name would be as follows and the script would point you to the folder in which the log file is created. The main installation script can be found in appendix A.

**CCDC_installation.log**

The script would exit if any of the requirements for application installation has not been met and the same information log would be available in the installation log. This information would be helpful in solving the problem which caused exiting of the script.
Chapter 5

USER GUIDE

This chapter provides an overview of what users can do with scoring interface. This section is intended for users or judges who conduct the game and part of white team. The user interface is designed to be simple and easy, which is required to have minimal learning to use application.

Figure 5.1 Login Page
5.1 Login Page

The application main page can be opened using the URL http://localhost/CCDC.php and the page asks for authentication. The authentication username and password are same as what we have for Nagios. Since the application interface needs to use same authentication as Nagios. Figure 5.1 shows the screen shot of the login page.

5.2 Select Team page

This page prompts user to select a team to show the scores. These options in the drop down are the names of the defender teams for which we provide scoring on a negative scale. These teams are the host groups in the Nagios instance. The team can be selected from the list provided and submit should be clicked to view the scores. The screen shot for the team select page is show below in Figure 5.2.
5.3 Score Page

This page gives the score for a particular team which is selected in the team selection page. There following are the details provided by the score page as shown in Figure 5.3 Team score Page

<table>
<thead>
<tr>
<th>Nagios Instance</th>
<th>Team Name</th>
<th>Points Deducted</th>
<th>Check Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Red1</td>
<td>6512</td>
<td>Click Here / Email Scores</td>
</tr>
</tbody>
</table>

Last Updated at 2008-11-18 12:00:04.

Figure 5.3 Team Score Page

- Nagios instance from which the data is collected
- Team name
- Points deducted due to down time
- Link to check details of the team i.e., which servers are maintained by the team
- Send scores to email link
5.4 Send Email Page

If the user selects the Email scores link in the Team score page the Send Email page is shown where the user can enter the email address of the team and click on **Send scores** button to send the email. If the application is unable to send the email due to any reason, the error is displayed on the same page. If the email is successfully sent to the team email address the page is re-directed to CCDC home page, where user can select another team and send score. This page enables the user to send the scores frequently.

![Figure 5.4 Email Scores Page](image)

Please enter email address

![Figure 5.5 Default Format of the Email Message](image)
5.5 Team Details Page

As explained in the section 5.3 from the team score page if the user clicks the “Click here”, the team details page will be displayed. The snapshot is provided in Figure 5.6

- Host Name of the server
- Link to summary graphs of each host
- IP address of the host
- Services hosted by the server
- Link to check the down time for each of the services hosted.
If the user clicks on the *host name*, the performance graphs for all the services as well as the host uptime will be displayed as shown in the Figure 5.7. This page contains only the information for each day. If the user needs to know the detailed view of the performance within a day, he has to navigate to the “View downtime history” link to check the details.

![Figure 5.7 Host Summary Page](image)

Figure 5.7 Host Summary Page

Figure 5.8 shows the snapshot of the histogram graph for the performance review. The x-axis of the day or time unit of the graph and the y-axis shows the percentage of time the service was up and running.
Figure 5.8 Performance Graph

5.6 Downtime History Page

This page helps user to answer questions from students if they want to know the downtime pattern for their services. Since points are deducted based on the SLA for downtime, doubts from students can be clarified using this page. User can navigate to this page by clicking “View downtime history” as shown in the Figure 5.6 Team Details page. The downtime history page can be divided into two pieces of information one is the time shown in minutes and the other a histogram graph which can be queried using the query form under the downtime information as shown in the Figure 5.9.
The page also provides two links in the top right corner for easy navigation.

- Link to application home page to select a different team
- Team Details page to navigate to a different service.

Figure 5.10 shows the information that can be viewed for the downtime history. It has details of name of the service, downtime starting time stamp, ending time stamp, number of minutes the service was down.
Downtime History for service DNS

<table>
<thead>
<tr>
<th>Start Time</th>
<th>End Time</th>
<th>Total Time (in Mins)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008-11-12 17:20:00</td>
<td>2008-11-12 17:25:00</td>
<td>5</td>
</tr>
<tr>
<td>2008-11-12 16:20:00</td>
<td>2008-11-12 16:25:00</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 5.10 Service Downtime Information

The section below the downtime history information has a query form where the user can enter the date range for which he wants to check the downtime. This page can show the performance graph in a detailed manner up to each hour of the day. Figure 5.11 shows the overall snapshot of the query form. User can enter the date range using the following drop down lists as show in the Figure 5.12. When user is done with date range selection he can press the submit button to get the histogram graph as shown in the Figure 5.13. We can see that a particular service is down from 6PM to 8PM on a particular day and the team would loose points for that downtime.
Figure 5.11 Downtime History Graph

Please enter the date range

From date: January 1, 2008

To date: January 1, 2008

Submit

Figure 5.12 Date Range Parameters
Figure 5.13 Downtime Performance Graph
Chapter 6

CONCLUSION

With simple user interface, graphical reports and facility to email scores to students, the project developed successfully meets all the requirements needed for scoring system for conducting cyber defense competition. The developed product along with Nagios [4] provides automatic down time scoring. The application is re-usable with the auto-installation script ready to deploy when ever needed. The application can be web deployed if necessary and has all features to answer the questions from students during the competition. The project will be useful in conducting such competitions in the future.

However, the future work would be to enhance the application by adding the following features:

- Enabling results of the downtime history page to be exported to Excel sheets.
- Developing a user privilege system, which can allow restricted access to students to check their status or score themselves.
- Web deploying the application to be accessible on any computer remotely.
- Adding background images to the web pages using a uniform color codes.
The application has been used to score the cyber defense competition on California state university, Sacramento campus. Based on the usage of the application, the following are the strengths and weaknesses of the product.

**Strengths of the Product**

- Product is easy to use with hyperlinks and buttons, user doesn’t need to enter data using keyboard anywhere in the application
- The product works with same efficiency with any number of teams participating
- The database configuration and database coding doesn’t need any changes even if there is change in configuration of Nagios [4]
- Auto installation script enables the product to be ready for deployment with minimum configuration

**Weaknesses of the Product**

- Application is not portable to any other operating systems, making it dependent on Centos operating system
- Distributed collection of data can cause lot of traffic in the network and also makes the data less secure
#!/bin/sh
#
#
########################################################################
############
#  Script Name: InstallConfigureDB.GUI.sh
#  Purpose: Installs and Configures NDOUtils, DB and GUI for CCDC scoring system
#  Author: Venkata Lakkaraju
#  Email: vijay.lakkaraju@gmail.com
#  Date created: 03/24/2009
#  Usage:
#  Start: No parameters required
#  End: Exits with status 1 if encounters error and
#       Creates CCDC_installation.log after the script stops
#       it contains installation log for this script
########################################################################
############

#Create a log file for installation
chmod 755 Configure.sh
touch CCDC_installation.log

#Store the log file name
CURR_FOLDER=`pwd`
LOG_FILE=CCDC_installation.log
echo ${LOG_FILE} > $LOG_FILE
echo 'Checking Dependecies before compiling' >> $LOG_FILE
#Check for php-gd dependency install if needed
if rpm -qa | grep php-gd >> $LOG_FILE
then
echo 'PHP-GD package installed' >> $LOG_FILE
else
echo 'PHP-GD package not installed trying to install ' >> $LOG_FILE
yum -y install php-gd >> $LOG_FILE
OUT=0
if $OUT -ne 0
then
echo 'Error in installing PHP-GD package ..exiting script. Please make sure PHP-GD package is installed' >> $LOG_FILE
exit 1
fi
fi

#Check for mysql-devel dependency install if needed
if rpm -qa | grep mysql-devel >> $LOG_FILE
then
    echo 'mysql-devel package installed' >> $LOG_FILE
else
    echo 'mysql-devel package not installed trying to install' >> $LOG_FILE
    yum -y install mysql-devel >> $LOG_FILE
    OUT=$?
    if $OUT -ne 0
        then
            echo 'Error in installing mysql-devel package ..exiting script. Please make sure mysql-
            devel package is installed' >> $LOG_FILE
            exit 1
fi
fi

#Check for mysql-clinet dependency install if needed
if rpm -qa | grep mysql-client >> $LOG_FILE
then
    echo 'mysql-client package installed' >> $LOG_FILE
else
    echo 'mysql-client package not installed trying to install' >> $LOG_FILE
    yum -y install mysql-client >> $LOG_FILE
fi

#Check if nagios is running
if /usr/local/nagios/bin/nagios -V >> $LOG_FILE
then
    echo 'Nagios is installed' >> $LOG_FILE
    if ps -A | grep nagios >> $LOG_FILE
        then
echo 'Nagios is running..' >> $LOG_FILE
echo 'Stopping nagios for installation..' >> $LOG_FILE
service nagios stop

fi

else
    echo 'Nagios not installed exiting the installation' >> $LOGFILE
    exit 1

fi

echo 'Inflate the compressed file' >> $LOG_FILE
tar xzf ndoutils-1.4b7.tar.gz

#Entering the NDOUtils folder
cd ndoutils-1.4b7
CHG_FOLDER=`pwd`

./configure --with-mysql-lib=/usr/lib/mysql >> $CURR_FOLDER/$LOG_FILE
OUT=$?
if $OUT -ne 0
    then
        echo 'Error in NDOUtils configure command check if mysql libraries are in folder /usr/lib/mysql' >> $CURR_FOLDER/$LOG_FILE
        exit 1

    fi

# run make
make >> $CURR_FOLDER/$LOG_FILE
OUT=$?
if $OUT -ne 0
    then
echo 'Error in NDOUtils make command .. exiting script .. please re-run InstallConfigureDB_GUI.sh' >> $CURR_FOLDER/$LOG_FILE
    echo 'Check CCDC_installation.log for information' >> $CURR_FOLDER/$LOG_FILE
    exit 1

#Do DB script

cd ..
    echo 'Checking if mysql is running before running script ...' >> $CURR_FOLDER/$LOG_FILE
    if ps -A | grep mysqld >> $CURR_FOLDER/$LOG_FILE
    then
        echo 'MySQL is running..' >> $CURR_FOLDER/$LOG_FILE
    else
        echo 'MySQL not running ..' >> $CURR_FOLDER/$LOG_FILE
        echo 'Starting MySQL ..' >> $CURR_FOLDER/$LOG_FILE
        service mysqld start >> $CURR_FOLDER/$LOG_FILE
    fi

    mysql < DBscript.sql >> $CURR_FOLDER/$LOG_FILE
    cat $CURR_FOLDER/$LOG_FILE
    cd ndoutils-1.4b7
    cd db
    CHG_FOLDER=`pwd`
    ./installdb -u nagiosdb -p nagiosdb -h localhost -d nagios >> $CURR_FOLDER/$LOG_FILE
    OUT=$?
    if $OUT -ne 0
    then
        echo 'Error in NDOUtils DB script.. exiting script .. please check CCDC_installation.log for information' >> $CURR_FOLDER/$LOG_FILE
        exit 1
    fi
fi

cd ..

#configuring the NDOMOD broker module

cd src
  cp ndo2db-3x log2ndo file2sock /usr/local/nagios/bin > $CURR_FOLDER/$LOG_FILE
  cp ndomod-3x.o /usr/local/nagios/bin/ndomod.o > $CURR_FOLDER/$LOG_FILE
  cd ..
  cp config/ndomod.cfg /usr/local/nagios/etc > $CURR_FOLDER/$LOG_FILE

  echo 'broker_module=/usr/local/nagios/bin/ndomod.o
  config_file=/usr/local/nagios/etc/ndomod.cfg' >> /usr/local/nagios/etc/nagios.cfg

#Installing NDO2DB Daemon

  cp src/ndo2db-3x /usr/local/nagios/bin/ndo2db > $CURR_FOLDER/$LOG_FILE
  cp config/ndo2db.cfg /usr/local/nagios/etc > $CURR_FOLDER/$LOG_FILE
  /usr/local/nagios/bin/ndo2db -c /usr/local/nagios/etc/ndo2db.cfg > $CURR_FOLDER/$LOG_FILE
  touch temp
  sed -e 's/ndouser/nagiosdb/' -e 's/ndopassword/nagiosdb/' /usr/local/nagios/etc/ndo2db.cfg > temp
  cat temp > /usr/local/nagios/etc/ndo2db.cfg
  rm temp

  echo 'Finished installing NDOUtils' > $CURR_FOLDER/$LOG_FILE
  cd ..

#compile the stored functions used in PHP pages

  echo 'installing functions for CCDC' > $CURR_FOLDER/$LOG_FILE
  mysql --database=nagios < statehistory.sql > $CURR_FOLDER/$LOG_FILE
  mysql --database=nagios < Calculatepoints.sql > $CURR_FOLDER/$LOG_FILE
mysql --database=nagios < CalculateServPerf.sql >> $CURR_FOLDER/$LOG_FILE
mysql --database=nagios < Calculateuptime.sql >> $CURR_FOLDER/$LOG_FILE
mysql --database=nagios < Calculatehost.sql >> $CURR_FOLDER/$LOG_FILE

#Configuration change in php.ini for include_path

echo 'Configuration change in php.ini for include_path' >> $CURR_FOLDER/$LOG_FILE
    echo 'Checking file php.ini ..' >> $CURR_FOLDER/$LOG_FILE
if -e /etc/php.ini
    then
        echo 'File exists...' >> $CURR_FOLDER/$LOG_FILE
    else
        echo 'file php.ini not found in /etc .. exiting' >> $CURR_FOLDER/$LOG_FILE
        exit 1
fi

touch temp
sed 's/;include_path = ".:/php/includes"/include_path = ".:/php/includes:/var/www/html"/' /etc/php.ini >> temp
cat temp > /etc/php.ini
rm temp

if -d /var/www/html
    then
        echo '/var/www/html found ...' >> $CURR_FOLDER/$LOG_FILE
    else
        echo '/var/www/html not found .. make sure the directory exists ..exiting script' >> $CURR_FOLDER/$LOG_FILE
        exit 1
fi

cp $CURR_FOLDER/webpages/* /var/www/html >> $CURR_FOLDER/$LOG_FILE
echo 'checking for file httpd.conf ..' >> $CURR_FOLDER/$LOG_FILE

if -e /etc/httpd/conf/httpd.conf
then
    echo 'found ..' >> $CURR_FOLDER/$LOG_FILE
    cat $CURR_FOLDER/http_config >> /etc/httpd/conf/httpd.conf
else
    echo 'Not found .. exiting script' >> $CURR_FOLDER/$LOG_FILE
    exit 1
fi

#restart httpd after configuration changes

echo 'restarting httpd after config changes' >> $CURR_FOLDER/$LOG_FILE
service httpd restart >> $CURR_FOLDER/$LOG_FILE

#restart nagios after all done

echo 'restarting nagios after all done' >> $CURR_FOLDER/$LOG_FILE
service nagios restart >> $CURR_FOLDER/$LOG_FILE

echo 'Please check CCDC_installation.log file for the install log in the folder'
${CURR_FOLDER}

service nagios stop >> $CURR_FOLDER/$LOG_FILE
rm -f /usr/local/nagios/var/ndo.sock >> $CURR_FOLDER/$LOG_FILE
/usr/local/nagios/bin/ndo2db -c /usr/local/nagios/etc/ndo2db.cfg >> $CURR_FOLDER/$LOG_FILE

service nagios start >> $CURR_FOLDER/$LOG_FILE

#End of autoconfig
BIBLIOGRAPHY


