XScore Documentation

Release 1.0

XScorers

May 05, 2010

CONTENTS

This is the documentation for XScore, a framework for hosting network security competitions.

CHAPTER

ONE

INTRODUCTION

XScore is the scoring server for Cyber Storm, a network security competition in which teams will attempt to *hack* each others networks. Teams are to provide a number of network services that they will be expected to maintain despite attacks from opposing teams. The effectiveness of these attacks will be the determining factor in deciding the winner of the competition.

The XScore system provides the following services:

- Checking the status of various network services provided by teams.
- Maintaining a running score of throughout the competition.
- Providing an administrative interface to manage competition related events.

1.1 Goal

The goal of this project is to provide a framework for not only Cyber Storm, but also for future network security competitions. It is hoped that the infrastructure provided by XScore will be useful and provide a solid foundation upon which Louisiana Tech, as well as other institutions, can use to host similiar competitions

1.2 License

CHAPTER

TWO

REQUIREMENTS

To implement the infrastructure needed for the Cyber Storm competition, XScore will need to provide two main services: network service checking and a point based scoring system to rank the opposing teams. This document describes the requirements of this system.

2.1 Network Service Checking

Throughout the competition, each team will be expected to maintain a number of machines that provide a set of network services. XScore will check the status of these services on a regular interval to determine if another team has successfully *hacked* or disabled this service. Support for checking the following services will be provided:

- HTTP - Hyper Text Transfer Protocol
- FTP - File Transfer Protocol
- MySQL - Database Server
- SSH - Secure SHell

2.2 Scoring

To determine a winner of the competition, XScore will need to provide a system that awards each team points based on the status of the services they are able to maintain. XScore is to provide an automated system that will check these services and then award the given team an appropriate number of points.

To establish a well-defined method concerning when points are awarded, we will define the status of a service as one of the following states:

1. Service is up and operates as expected.
2. Service has been successfully hacked by an opposing team.
3. Service is down or does not operate as expected.

Each of these states will be recognized by XScore and will be the basis on which points are awarded. The value of the points can be specified for each of the services checked and configurable as needed.

In addition to this process of checking services and awarding points, XScore is to provide the officials of the competition's with the ability to manually adjust the scores. This is intended to provide a fallback in case if the scoring server incorrectly awards a team points. In addition, this will allow the officials to handle unforeseen events during the competition. For example, if a team attempts to *hack* XScore and modify the scores, officials may want to remove these points.

2.3 User Interface to XScore

To allow officials to easily run the scoring server and manage the compeititoin, XScore will provide the appropriate software. Running the scoring server will be handled by a command-line program and will schedule when and how often services are checked. To monitor the status of the competition and provide a friendly user interface to the scoring server, a web-based application will be included. This program can be used by officials to create new challenges and announce important messages. In addition, this program will provide the officials with access to manually adjust the scores as needed.

CHAPTER

THREE

DESIGN

The XScore system is divided into multiple components in order to provide a modular design that is easy to extend while also satisfying the neccessary requirements of Cyber Storm. This document is intended to provide an overview of how XScore is designed and implemented.

3.1 Overview

XScore consists of two main components: the scoring server and a visual front-end. The scoring server is responsible for scheduling when services are checked and the registering of the appropriate scoring events. In addition, the scoring server handles passing data to the scoring client via an HTTP server.

The visual front-end is a web-based application that provides an adminstrative interface to the scoring server. It provides the ability to monitor the progress of the competition as well manage various competition related events.

3.2 User Interfaces

3.2.1 Scoring Server

The scoring server is the command-line program `xscored` which is used to initialize and run the server.

3.2.2 Front-end

3.3 Programming Environment

XScore utilizes a number of different software systems to implement the scoring server. Although primarily written in Python, it also makes use of Bash shell scripts and the front-end makes extensive use of Javascript. Since this system is intended to be used on Linux based servers, little concern was given to portability on non Posix systems. This allowed us to simplify the implementation of the scoring server and prevented us from having to test on other platforms. It does however imply that additional development would be required if this feature were wanted in the future.

Other systems used and required by XScore include a MySQL database server and a HTTP server used to communicate with the scoring client and host the front-end. Refer to the README file included with the distrubution for the specific versions of the programs used by XScore.

3.4 Scoring Server

The scoring server is divided into four main components:

1. Network Service Checkers.
2. Scheduling of Service Checks.
3. Scores and Events Handeling.
4. Communication with the client.

These components are discussed in the following sections.

3.4.1 Network Service Checkers

The network service checkers are used to determine the status of a team's network service.

FTP

HTTP

SSH

MySQL

3.4.2 Scheduling of Service Checks

3.4.3 Scores and Events Handeling

Record and manage the scores for each team and the events that occur during the competition.

3.4.4 Communication with the Client

CHAPTER

FOUR

XSCORE

The `xscore` package implements the routines used by XScore to check network services and handle new scoring events.

4.1 checkers

Module that handles checking the current status of SQL, FTP, SSH, and HTTP services for each team.

class FTP (*team, ip, port, usr=None, passwd=None, timeout=30, koth=False*)
FTP Service class

get ()
Attempt a connection to FTP server and return welcome message from server.

class HTTP (*team, ip, port, usr=None, passwd=None, timeout=30, koth=False*)
HTTP Service class

get ()
Connects to http URL and returns server response.

class MYSQL (*team, ip, port, usr=None, passwd=None, timeout=30, koth=False*)
MYSQL Service class

get ()
Attempt a connection to MYSQL server and return list of all databases.

class SSH (*team, ip, port, usr=None, passwd=None, timeout=30, koth=False*)
SSH Service class

get ()
Attempt a SSH connection to ip and return MOTD from server.

class Service (*team, ip, port, usr=None, passwd=None, timeout=30, koth=False*)
Service class extended by all Network Services, contains facilities to check status of itself and format a response message.

check ()
Check the status of this service.

Returns the status and a descriptive message.

format (*fmt=None*)
Construct a fancy message describing this service.

'fmt' can contain the following %-style mapping keys which are replaced with the appropriate values.

%(team) Team name %(ip) IP address %(port) Port %(service) The service checked %(status) Status of the service (UP | DOWN | PWNED) %(hacker) If 'status' is PWNED this contains the hacker's name. %(reason) If 'status' is DOWN this describes why.

stat()
HTTP Service class

Get the status of this service.

Returns 'UP', 'DOWN', or 'PWNED'

exception TimeOutException
Triggered when a service takes too long to respond.

check(*team, service, ip, port, usr='', passwd='', timeout=30*)
Checks the status of a service and appropriates points accordingly.

timeout(*seconds, func, *args, **kwargs*)
Executes the function 'func' with the given arguments. If it does not return within the time specified by 'seconds', a TimeOutException is raised.

4.2 scores

Database Facilities to handle creation and management of events, announcements, and challeneges.

add_announcement(*msg, secs_to_display=10*)
Add a new announcement to the database.

add_event(*team, etype, pts, msg*)
Submits new event to database.

end_challenge(*id, winner*)
Declare a winner for an existing challenge.

new_challenge(*challenge_name, pts, msg*)
Create a new challenge.

query(*sql, sql_args=None*)
Executes query and returns resultset.

4.3 logger

Facilities for logging all activity generated by the XScore Server, including service status checks and database manipulation.

4.4 config

Contains all neccissary configuration data for each teams servers as well as all neccissary scoring values for the competition.

CHAPTER

FIVE

NAME

5.1 SYNOPSIS

5.2 DESCRIPTION

5.3 OPTIONS

5.4 EXAMPLE

5.5 SEE ALSO

CHAPTER

SIX

NAME

6.1 SYNOPSIS

6.2 DESCRIPTION

6.3 OPTIONS

6.4 EXAMPLE

6.5 SEE ALSO

CHAPTER

SEVEN

NAME

7.1 SYNOPSIS

7.2 DESCRIPTION

7.3 OPTIONS

7.4 EXAMPLE

7.5 SEE ALSO

CHAPTER

EIGHT

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*

MODULE INDEX

X

INDEX