SECURITY REPORT

PREPARED FOR DAYTRADING INC

PROJECT NAME:	*DAY TRADING SOFTWARE*
SYNOPSIS:	*AN EXPLANATION OF THE SECURITY DESIGN FOR THE PROJECT, THREAT MODEL, AND A DESCRIPTION UNTRUSTED DATA SOURCES.*
DATE:	
PREPARED BY:	*BRAIGHTON POLACK (V00763687), EVAN WILLEY (V00703788)*

CONTENTS

1. INTRODUCTION

The purpose of this document is to outline the sources of potentially malicious data in the day trading system, and to describe the measures used to reduce possible threats. The way data is handled at each component is described in detail below.

2. SOURCES OF UNTRUSTED DATA

The two primary sources of untrusted data in the system come directly from either the user input from the web interface, or the quote responses from the quote server. User input going to the web server is the most important data to sanitize. If left untouched, this input could be used to insert malicious code into the web applications. Another potentially unsecure area is the connection between the client and web server, as ideally this is the only part of the system exposed to any external network. Denial of service attacks could also be an issue for any components connected to the internet, though the use of multiple web servers, and purchasing a higher bandwidth connection could alleviate these issues.

The figure below illustrates the potential threats to the system caused by untrusted data.

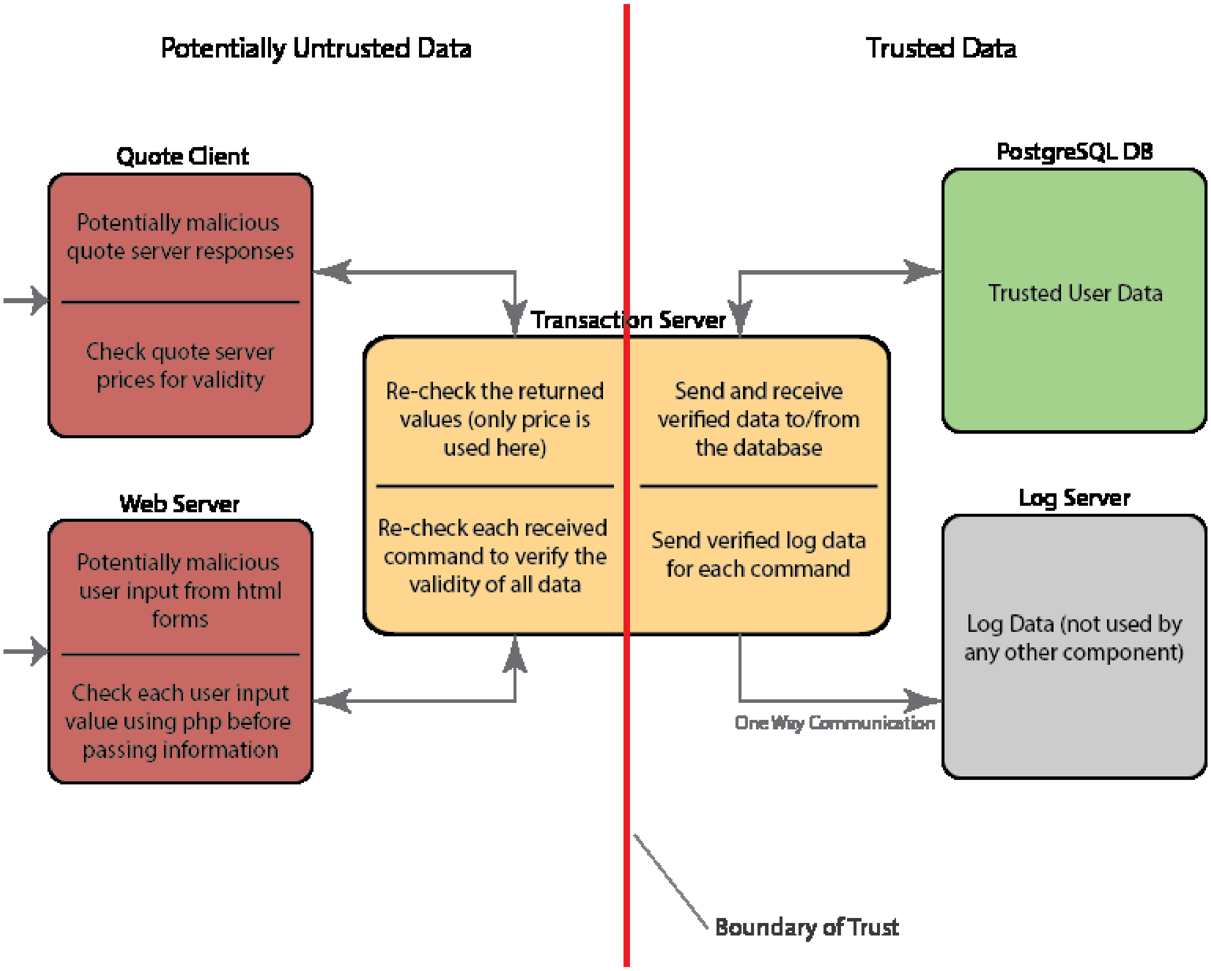
Potentially Untrusted Data
Trusted Data
Quote Client
Potentially malicious quote server responses
Check quote server prices for validity
Web Server
Potentially malicious user input from html forms
Check each user input value using php before passing information
Transaction Server
Re-check the returned values (only price is used here)
Re-check each received command to verify the validity of all data
Send and receive verified data to/from the database
Send verified log data for each command
PostgreSQL DB
Trusted User Data
Log Server
Log Data (not used by any other component)
One Way Communication
Boundary of Trust

FIGURE 2.1 - THREAT MODEL FOR DAY TRADING SYSTEM

3. Countermeasures by Component

The measures taken to remove malicious data; and to protect the system from attacks in general, are listed below. There is no guaranteed way to be protected against every kind of attack, but a proper incident handling procedure could be implemented to minimize issues in the case of a successful attack.

3.1. LOAD BALANCER

The quote server is a source of untrustworthy data; as it could either return incorrect data, or become compromised and begin sending malicious data. To counteract this, the quote client checks data received from the server. In particular the price is checked to be numeric and above zero. If the price is invalid, the quote request will not be logged, though the response will still be sent to the transaction server.

3.2. WEB SERVERS

The web servers handle data that was directly input from a user into the html forms on the web interface. These html forms have some restrictions on them as to what can be entered, such as the numeric input types only allow numbers to be entered. These forms are not entirely secure, and it is very likely that a user could get around the html restrictions quite easily. For example, some of the inputs do not restrict input if the user simply copies an entire string into the text box.

To counteract these security flaws, each string or value input into an html form is checked on the web server using php. The server removes all special characters from any values, and checks the length of each value to ensure the correct number of characters. The following characters are allowed for each input type.

- Usernames allow upper/lowercase alpha-numeric characters
- Stocks allow upper case alphabetic characters
- All prices and amounts must be numeric with value greater or equal to 0.01
- All prices are automatically rounded to the nearest cent
- Symbols such as commas are not permitted inputs, though commas are generated automatically for monetary values

In a full implementation of the system, the connection between the client and web server would be encrypted with SSL. This would ensure that when a complete username / password specific system is created, data is kept private. This would also obscure any command data an active user sends to the system.

To further secure user input, a white-list system could be followed. In this approach, only inputs matching predefined patterns would be allowed through, and all others filtered out.

3.3. TRANSACTION SERVERS

The transaction servers could potentially receive untrusted data that is missed by the web servers or quote client. In this way it is the last defense in the system against malicious data. Before performing any command, the transactions server double checks the validity of every variable it has received.

The transaction server checks each price and dollar amount to ensure that each is numeric and above zero. It also checks most strings (relevant to the command), particularly user IDs and stock symbols for length and characters. It also checks the total message length of each received command, which essentially is the number of variables received in a delimited format. If the length is too short or too long for that command, it is rejected.

This double checking is important for both the web server data, and the quote client data. The quote client itself does not check certain things such as message length, so checking it on the transaction server is required. The data received from the web server must be double checked, because the web server uses php it is open to some vulnerability if a code injection attack were to be successful.

4. CONCLUSION

In conclusion, there are several sources of untrustworthy data entering the trading system. Each of the components on the left side of the figure in section 2 has a method of filtering this data to remove anything malicious or incorrect. Though it is impossible to guarantee security from all attacks while connected to unknown data sources, these methods make the system fairly secure.