MITRE Baseline Configuration System

Software Requirements Specification

FINAL REVISION, September 17, 2008
Purdue University, CS 307, Fall 2008

Team MITRE:
Catherine Brown
Michael Dunn
Mark Nowicki
David Tittle

1. Introduction

1.1 Purpose

This document contains the preliminary Software Requirement Specification for the MITRE Baseline Configuration System. It contains a description of the project, a detailed list of features offered as well as use cases and other nonfunctional requirements. This document will also provide a working description of the MITRE team's project in its entirety and should work as a reference for the MITRE team, class supervisors, and the corporate partners.

1.2 Overview

This Software Requirement Specification will document all of the features and use cases that the team will compile over the course of this project. It also provides a guideline for the creation of user manuals and other useful documents.

2. Description

2.1 Project Synopsis

The MITRE program will be a running script that will retain, compare, and parse information from provided XML files. The application will need to be able to interpret various XML schemas such as Nessus and SCAP. The program will run on a single server and will securely transfer XML files from provided resources. Access to the script, its configuration files, its XML files, and its database must be limited and secure. The program will also be able to alert personnel via email of inconsistencies in the XML files. A self-diagnostic tool will be included to ensure its accuracy.

2.2 Project Domain

The MITRE Corporation is a not-for-profit organization chartered to work in the public interest. MITRE is a Federally Funded Research and Development Center (FFRDC) that analyzes technical questions with a high degree of objectivity, and provides creative and cost-effective solutions to government problems. Because of the open source requirements, the project will initially be working within the domain of the MITRE Organization, but in the future might fall within the domain of other organizations as well.

2.3 Project Environment

The software should be platform independent, and able to work in various environments, especially UNIX environments.

2.4 Project Platform

The software will be written in an open source language. It will incorporate an open-source database tool. It also needs to be open source for maintainability, reusability, and extensibility. Open source is also important because MITRE is a federally funded organization and this application may eventually be released to the public.

2.5 Constraints

- The software will run on a single server that will be dedicated to running the software.
- It will be able to retrieve XML documents by using a standard transfer protocol.
- Access to the software will be limited to those users who have the proper permission to access the client configuration data. Each user can have a different level of permission.
- The software will need to adapt to various XML formats.
- The software solution will need to be as independent as possible from libraries or environment system resources to ensure its ability to run without interruption or failure.
- The software will include instructions necessary to administer and install the software, including a detailed explanation of the various configuration files used to create users and set appropriate permissions.

2.6 Dependencies

This project assumes that the server can handle the load placed on it by the software and is not restricted by any other software or hardware to complete its assigned functions. It is also assumed that the server executing the software is physically secure and that the child servers and XML files will be available for file transfer over the network. This project assumes that correct configuration files will be provided.

3. System Features

3.1 Automated Document Retrieval

The software will be able to run on a single main server, using an automated document retrieval process. The server will establish a connection with any child servers via the network, and initiate an XML file transfer using a standard transfer protocol.

3.2 XML Parsing Functionality

The software will be able to parse collected XML files while comparing to previously collected files in order to detect changes throughout a file's history.

3.3 XML Format Flexibility

The software will have the capability to be configured to scan any kind of XML schema. Different XML schemes can be added or removed from the scanning process depending on the needs of the system/user.

3.4 Database Compatibility

The software will be able to add, update, and remove information retrieved from XML files into a database. A version history will also be stored in the database, which will be used when doing version comparisons as well as maintaining a baseline for these comparisons. This will be done using an open source database technology.

3.5 Configurable Update Alerts

The software will allow users to determine which changes discovered in the XML scans/comparisons will be alerted, ignored, or documented. Users will be able to determine who is alerted when they are confronted with different types of changes in configuration.

3.6 Modifiable Baseline History

The software will allow users to configure baselines for each specific child server on the network. Users can make any modification in the history (which is stored in a database) a baseline for future comparisons.

3.7 Access Control

The software will allow the system to discriminate on whom uses the client interface module to access any of the child server histories. It will use a username/password system that will use a previously defined Active Directory.

3.8 Shell Access

The software will allow the system to be accessed from Network Administrators and/or Security Managers to view XML version history, system status, and to enter command line functions. The Command Line functions can be used to manually trigger automatic system events such as scanning and self diagnostic routines.

3.9 Dynamic Client Handling

The software will allow the system to efficiently and effectively maintain a list of active server connections. Alerts will be sent if a new server becomes active, or an existing server is unable to transfer the XML file to the main server due to a connection loss.

3.10 Self Diagnostic Functionality

The software will allow the system to monitor its current 'health' with the use of test scans/comparisons and weekly alerts notifying users of current health situation. It keeps a log file that has weekly health updates logged into the file. It can only be accessed by credentialed users.

4. Use Cases

For the current use cases, the **actor** is the **server** and the **system** is the **XML Baseline Tool** unless otherwise noted.

4.1 Use case: Fetch XML Files

ACTOR ACTIONS	SYSTEM RESPONSES
1. Trigger XML file fetching	2. Transfer protocol initiated
	3. Access server directories
4. Store files	

4.2 Use case: Check XML Format

Related use cases:
Includes: Parse XML File

ACTOR ACTIONS	SYSTEM RESPONSES
1. Trigger XML format check	2. Load initial lines of XML file
3. Parse XML File	

4.3 Use case: Parse XML Files (inclusion)

ACTOR ACTIONS	SYSTEM RESPONSES
1. Trigger parse XML files	2. Request file creation date
3. Provide date	
4. Choose format check or full parsing	5. Parse file
	6. Add parsed information to database

4.4 Use case: Store History (Updates)

Related use cases:
Includes: Send Alerts, Check XML Format

ACTOR ACTIONS	SYSTEM RESPONSES
1. Prepare (open up) database	
2. Trigger store history	3. Query the database
4. Create history file	5. Update history file
	6. Read tag priority configuration file
7. Check XML Format	8. Locate high priority changes
9. Send Alerts	

4.5 Use case: Send Alerts (inclusion)

ACTOR ACTIONS	SYSTEM RESPONSES
1. Trigger send alerts	2. Locate and pass on e-mail addresses
3. Send e-mail alerts	

4.6 (Human User) Use case: Modify Baseline

ACTOR ACTIONS	SYSTEM RESPONSES
1. Choose 'Baseline' command	2. Request server name
3. Input server name	4. Request workstation name
5. Input workstation name	6. Retrieve baseline configuration file
	7. Display baseline configuration file
8. Input/Choose new baseline date	9. Display updated configuration file and await confirmation
10. Confirm update	11. Save configuration file and exit mode

4.7 (Human User) Use case: Remove Entry from Database

ACTOR ACTIONS	SYSTEM RESPONSES
1. Choose 'Edit Database' command	2. Display 'Edit Database' menu
3. Choose 'Remove Entry' command	4. Display database entries
	5. Request entry to remove
6. Input entry number	7. Remove entry and request confirmation
8. Confirm	9. Save database and exit mode

4.8 Use case: Add Client Computer

ACTOR ACTIONS	SYSTEM RESPONSES
	1. Check baseline configuration file for workstation
	2. Workstation not found
3. Trigger Add Client Computer	4. Add workstation to configuration file
5. Send Alerts	

4.9 (Human User) Use case: Remove Client Computer

ACTOR ACTIONS	SYSTEM RESPONSES
	1. Check baseline configuration file for workstation
	2. Workstation found but no file present
3. Send Alerts	4. Prompt for workstation removal
5. Respond	6. Remove workstation from configuration file and save and exit or save and exit without removal

4.10 Use case: Make Log Entry

ACTOR ACTIONS	SYSTEM RESPONSES
1. Server becomes aware of application Malfunction	2. Write error entry to log file

5. Nonfunctional Requirements

5.1 Platform Independence

The software should be as flexible as possible. By using open source solutions, we enable the software to adapt to almost every platform and environment without any cumbersome licensing restrictions.

5.2 Secure Design

The software will need to be designed with security in mind. Not only does the network need to be kept secure, but the information that the application collects and retains also needs to be kept secure. To avoid security holes, all inputs and parameters to the application must be validated. This will ensure that unwarranted database query injections are protected against. Also, the software will need contain an access control system which validates the user's identity and the integrity of incoming commands.

5.3 Quality

The MITRE Baseline Configuration System will employ high standards of quality. The software will be designed in such a way that it can be maintained by another software team after deployment. With this, the software must have a high level of readability. This includes a strict format of white spacing that will be consistent throughout the entire software package. Also, this includes extensive commenting such as function descriptions, how they are used by other functions, function inputs, and resulting outputs. Availability is also key to the success of this software. The software will be needed to be running twenty-four hours a day, seven days a week to ensure a secure system. Several functionalities will be needed to be executed during non-scheduled scan times, thus the need for 24/7 availability.

5.4 Extendibility and Maintainability

The software should contain a high level of extendibility. Throughout the development of the application, the fact that the application will need to be used according to different standards and schema needs to be taken into account. By providing this level of extendibility, the software will be much more maintainable because of the aforementioned ability to adapt to different standards if the application is used with different input styles.

5.5 Reliability

The MITIRE Baseline Configuration System will function with a high level of reliability. Combined with previously defined system health checks and library functionality, all system errors will be handled preemptively to ensure stable processing.