SMARTECH

NetCheck Project
Test Specification Report

Neslihan Bulut

Kezban Demirtaş

Hande Çelikkanat

Gülşah Karaduman

Filiz Alaca

Department of Computer Engineering

METU

April 2006

TABLE OF CONTENTS

1. INTRODUCTION

Since we have started the implementation phase of NetCheck, we are trying our implementation to be:

- Error free,
- Complied with the project design,
- Complied with the project requirements.

In order to achieve this, it is very important for us to apply tests and detect any disharmony. With this in mind, this document is dedicated for explaining Smartech's test specifications while developing NetCheck project.

1.1. Goals and Objectives

The NetCheck project of the Smartech group is a web-based application level gateway project and should be implemented in a very limited time interval. Not only we have to implement the individual modules, but also we have to make all the modules work in accordance to form the whole system. At this point testing becomes one of the most important issues to detect any disorder from the beginning and correct those disorders in time.

Our goals while we are testing NetCheck are the following:

- Demonstrating that our product is error-free,
- Finding errors and correcting them in time,
- Ensuring that the customer requirements are met,
- Ensuring that the product provides all of the necessary functionalities,
- Ensuring the performance of the product.

1.2. Statement of Scope

Smartech's testing process for NetCheck consists of unit testing, integration testing, validation testing, and higher order tests.

Unit testing is applied for each module and the one who implements the module makes the necessary tests.

Integration testing is one of the most important testing concepts for our project. Most of the modules of NetCheck work in a thread. As a result the accordance between the modules gains more importance to prevent deadlocks and unexpected results.

Validation testing for NetCheck will be including requirement and design validation categories which are explained in detail in the following parts of this document.

The higher order testing for NetCheck includes performance tests, stress tests, and alpha and beta tests. The details will be explained briefly under the corresponding section.

1.3. Major Constraints

1.3.1. Time

Since NetCheck is an Application Level Gateway through which the network traffic flows, performance issues gain more importance. Group members spend time both on development and performance issues. Since improving performance is a very time consuming issue, time becomes the major constraint in testing the code.

1.3.2. Staff

The project is being developed by a team of 5 people. Since, these 5 people are working on both development and debugging, staff is one of the major constraints for the test.

2. TESTING STRATEGY AND PROCEDURE

It is important to mention here that this part of the document describes the testing strategy of Smartech and the procedure followed for testing our product, NetCheck. The following subsections explain the different testing procedures that will be used while testing NetCheck.

2.1. Unit Testing

According to Smartech's unit testing strategy, every developer should test the module that she has implemented. The point that our team considers while applying unit tests is that we firstly make sure that testing will be useful and it is not time consuming.

All of the modules will be tested in a black-box manner, by observing the input that is provided to the module and the output or the effect produced as a result. Output or effect results of modules can be explained by an example as follows:

The statistics module of NetCheck produces some diagrams and percentage analysis as output whereas the restriction module produces the effect of Internet access limitation.

Until this time, we have tested network traffic monitoring module, download and URL restriction modules, and firewall module which have been completely implemented. Furthermore, we have tested the implemented parts of content filtering and statistics module.

From now on, our plan is to implement logging and auto update modules and to apply unit testing to these modules before they are integrated to the system. We will also be improving the content filtering and statistics modules and applying unit tests to the newly implemented parts.

2.2. Integration Testing

Our project, NetCheck, is composed of different modules that have to interact with each other in order to form the whole system. There may be cases such that the module works correctly when it is independent from the system but it may not produce the expected

results when it is integrated to the system. To make sure that all of the modules work together correctly we have to define a testing strategy for the integration of modules.

Smartech has decided to apply bottom-up integration testing. Modules that have been tested by unit testing will be added to the system one by one and it will be tested whether whole system works correctly after the addition of the new module. Furthermore, we will be testing whether the added module still works correctly and does its job in accordance with the already integrated and tested ones.

Until now, we have integrated network traffic monitoring module, download and URL restriction modules, firewall module, and the implemented parts of statistics and content filtering modules. We have added those modules one by one to the system and observed whether they are still working correctly.

From now on, we will be integrating logging and auto update modules to our system and testing the behavior of the system as a whole. Moreover, when a module is improved independently from the other ones, the improved version will be integrated and tested by following the same strategy.

2.3. Validation Testing

In the scope of NetCheck, validation tests are also very important. In this type of testing our aim is to understand whether our product meets the requirements and complies with the expectations of the customers. Smartech has divided the validation test strategy and procedure into two subgroups as the following.

2.3.1. Requirements Validation

Requirements validation tests will be held in a black-box manner. In this type of testing we will be making use of our analysis report that we have generated according to the customers' expectations and requirements. In our report, we have mentioned about the functionalities of the system as follows:

- Filtering content of incoming packets from black words,
- Restricting access to some predefined sites,
- Taking logs of network traffic for later inspections of the administrators,

- Calculating statistics for administrators to gain insight about the network,
- Monitoring the current status of the network,
- Ability to manage access and download rights for groups of users and/or individual users.

During the requirements validation, we will be testing whether all of the above functionalities are provided by NetCheck.

Until now, we were able to test the validity of individual modules which were fully implemented. Moreover, we have tested NetCheck's first release package. In the coming days, we will be completing the implementation of NetCheck and testing the whole system in terms of requirements validation.

2.3.2. Design Validation

Smartech has the aim of implementing NetCheck nearly fully consistent with its final design. For this purpose, during the implementation of each module and class we make use of our final design report. White-box testing strategy is used in order to test design validation.

Until now, the implemented modules have been validated in terms of design. We will also be testing the design validation of the newly implemented modules in the following times.

2.4. Higher Order Testing

Besides unit, integration and validation testing, we will perform some higher order testing. Since our product will manage overall network traffic of a company, performance tests and stress testing gains more importance.

2.4.1. Performance Testing

NetCheck, being an Application Level Gateway, will manage the overall network traffic of a company. All incoming and outgoing web traffic will be handled by NetCheck, so performance testing is a must. When a user wants to connect to the Internet, he/ she should not wait for a long time to reach the web site he/ she wanted. That is, our program should do its job in a very short period of time. In order to test this, we will firstly request

many sites from one user. After being successful in this test, we will request many web sites from many numbers of users. Our system should be able to handle the network traffic when there are lots of requests at the same time.

2.4.2. Stress Tests

We will test our program for abnormal situations. When there are too many tasks to be completed, the program should not crash and handle the situation. We will send so many requests concurrently from so many users, and the program should not be locked. Moreover, we will be testing our system in such cases where more than one administrator are changing the configuration of the system at the same time.

2.4.3. Alpha and Beta Testing

Before first release of our product, we integrate all of the completed modules and prepare the product for alpha testing. Our alpha testing schema includes testing the code using white box testing. Also, we plan to perform additional inspection by using black box testing. After completing the alpha test, the product is in the form that it can be presented to the end user. We plan to release our product to a limited audience outside of our company so that further testing can ensure the product has few faults or bugs. With the feedback from our audience, we can both correct the faults and improve the program performance.

3. TEST RECORD KEEPING AND LOGS

During the testing phase, we usually aim to test different perspectives of the program. Also, our product being an Application Level Gateway, in testing, we have no specific output. To handle this, we keep a 2-column table in a word document, the first column is the tested property, and the second column is successfulness.

Tested Property	Successfulness
User based URL restriction	**Successful** (URL is restricted for the specified user; he/ she can not reach the requested web site. Web site is blocked via the firewall module.) **Unsuccessful** (User can reach the requested web site although URL is restricted for the user.)
Group based URL restriction	**Successful** (The requested URL is blocked for all of the group members for the specified group) **Unsuccessful** (The requested URL is not blocked for all of the group members for the specified group. It behaves undeterministic.)

With the help of the records that we keep, we will be able to see where the bugs exist in the program.

In a similar manner, we will be keeping a 4 column log table in the following format:

Tested Property	Explanation of the bug found	Date of bug detection	Date of bug correction
User based URL restriction	User can reach the requested web site although URL is restricted for him / her.	19.04.2006	22.04.2006

4. TESTING TOOLS AND ENVIRONMENT

The following testing tools and environments will be used during testing:

- g++ (for Linux)
- gdb (for Linux)
- Mozilla Firefox
- Internet Explorer
- Apache (for testing the Web part of the project)

5. TEST RESOURCES AND STAFFING

We will be using no further resources other than our development environment for testing purposes.

The staffing strategy of Smartech for testing is as follows:
We do not have any specialized tester in our team. That would be a luxury for us to make one of the team members work as a tester because our team consists of only 5 members and all of the members are assigned many tasks other than testing and all of those tasks should be completed in a very limited time. For handling this situation, we decided to make all the members to participate in the testing process and test the modules that they have implemented. Whole system testing is handled by all of the members.

6. TEST SCHEDULE

The following table shows our testing schedule.

Task	Handled by	Starting Date	Ending Date
Test Specification Documentation	Gülşah-Filiz	17.04.2006	23.04.2006
Unit Testing of Modules	All Members (Each member tests the modules that she implemented)	06.03.2006	07.05.2006
Validation Tests	All Members	20.04.2006	14.05.2006
Performance Tests	All Members	01.05.2006	21.05.2006
Alpha Tests	All Members	20.04.2006	24.05.2006
Beta Tests	Audience	14.05.2006	24.05.2006
Stress Tests	All Members	20.05.2006	26.05.2006