Software Test Plan

TauNet

Josh Bucklin
CS300

Table of Contents

1.0 Purpose:

This document outlines the testing strategy I used when testing the main components of the TauNet software. The tests were performed to prove to a reasonable satisfaction the main components of TauNet work as intended. The components include:

1. CipherSaber-2/RC4 located in the file (cs2.py)
2. TauNet Server located in the file (TN_Server.py)
3. TauNet Client located in the file (TN_Client.py)

1.1 Testing Strategy:

Due to the nature of the software's reliance on the internet and the complexities that can occur with it my testing strategy is largely focused on acceptance testing where possible. In addition to this, unit tests will be conducted for CipherSaber-2 to ensure that it is producing appropriate output. Finally I will perform tests that ensure the software can interact with other users of TauNet network.

It is expected that at a minimum all functional & non-function requirements outlined in the SRS will be met during this testing plan.

2.0 Testing:

This section is broken down into each of the main software components and the tests I performed on each.

2.1 CipherSaber-2/RC4:

My approach to testing this portion of the software was to build a testing program that loads different text strings, encrypts them and then decrypts them ensuring that the text matches at the start and end. In addition to this I also did fail testing to make sure that if the key or the number of rounds of key scheduling were different that the message wasn't decipherable. I included the testing program in the file cs2_test.py. All the tests focus on input-output pairs and a sample of them is included below: (Only a few samples are included to save space.)

PASSING TESTS	FAIL TESTS - THESE ARE SUPPOSED TO FAIL
String Encryption:	Test - Different keys
String After Decryption:	String Encryption: This is a test string!
Test: PASS	ﾒ P���� A��"J�becryption: #�'��##
String Encryption: abc	Test: FAIL
String After Decryption: abc	Test - Different Rounds
Test: PASS	String Encryption: This is a test string!
String Encryption: 1_*()()$*	String After Decryption:
String After Decryption: 1_*()()$*	#�##�n�,n�{�S�A�2.���d
Test: PASS	Test: FAIL

These are a few other tests I performed on this portion of the software:

2.1.1 **Test:** Size of encryption should be size of message + iv length
Description: Another verification of Ciphersaber-2 is to verify that it outputs the correct length of messages. The TauNet Protocol v0.2 specifies a max message length so it's important that Ciphersaber-2 behaves in a know fashion. The tests input is a string which is then encrypted and decrypted. All sizes are displayed. The expected output of this test is that encrypted is 10 bytes bigger than both the plain text and decrypted.
Input: The string 'This is a test string'
Output: The number of bytes of the input plus 10 additional bytes for the iv length
Result: PASS

2.1.2 **Test**: Other TauNet implementations can decrypt messages from my TauNet
Description: If the encryption is performed properly other implementations of TauNet will be able to decrypt them. The focus of this test is to send messages to other TauNet users and Bart's server to make sure they can decrypt them.
Input: Send the message 'This is a test message' to another TauNet Node.
Output: Confirmation from the other node that they received and successfully decrypted the message.
Result: PASS – This test was completed with both the echo server and with another student's implementation.

2.1.3 **Test**: My TauNet node can decrypt messages from other TauNet nodes
Description: Another user and I agree on a known string. That user sends me the message and it is displayed on my screen. Then I will visually verify that it's the agreed upon message.
Input: The encrypted message from another TauNet node.
Output: The decrypted message on the screen.
Result: PASS – I achieved this result by testing with my own implementation and another student's implementation.

2.2 TauNet Server:

The testing approach for this portion of the software is largely acceptance testing based on the SRS document. Again this is done because of the complexities of TCP/IP and the protocol. However, there are tests aimed at ensuring the server complies with TCP/IP and the TauNet protocol. Desk checks were performed on all code developed for this section.

2.2.1 **Test**: The program can take input from the keyboard and is display it on the screen
Description: The SRS specifies that the user can interact with the software via a keyboard/monitor. This test verifies that's true.
Input: Once the server is loaded. Type view and enter.
Output: The text entered ('view') will be displayed on the screen and the resulting information from the function call is displayed on the monitor.
Result: PASS

2.2.2 **Test**: The program can receive messages from other users
Description: According to the SRS document TauNet can receive messages from other TauNet nodes. This test receives input from another node and confirms that message is displayed on the screen.

Input: A connection request is received from another user.
Output: The request is accepted, the message is transmitted, then decrypted and displayed on the screen.
Result: PASS

2.2.3 **Test**: Only one function prints to the screen at a time
Description: Due to the fact the TauNet Server utilizes multiple threads it's important that no two functions write to the screen at the same time. Locks are used to prevent this from happening. This test verifies their implementation is correct by entering a prompt state waiting for input from the user and sending messages to the server and making sure they aren't displayed until the user proceeds.
Input: Prompt state is entered. In this case the prompt waiting for a new key. Then three messages are sent to the server.
Output: Once the new key is entered. Then all three messages will be displayed.
Result: PASS – The order of the messages is not guaranteed because threads seem to be unpredictable but the system does wait to display them.

2.2.4 **Test**: The server can receive messages up to 1KB in length
Description: The TauNet protocol specifies that the user should be allowed to enter a message of up to 1KB in length. This implies that the server will be expected to receive messages up to that length. This test is performed by opening the client, sending a max length message and making sure it's all received. Max message length is = 1KB – Total Header – IV Length.
Input: A message is received of 1KB in length.
Output: The entire message is properly displayed on the screen.
Result: PASS

2.2.5 **Test**: Warn user if an unknown user is sending them a message or if a different version of TauNet is being used.
Description: The SRS specifies that if an unknown user is sending a message or if a user receives a message from another node that is using a different version of the TauNet protocol a notification will be displayed and the message will be displayed after that.
Input: A message is sent from a user not in the list of known IP's and/or a message is sent from a node using a different version of the TauNet protocol.
Output: A warning is displayed on the screen.
Result: PASS – This test only verifies that the version # is the same not that all the headers are in the right place.

2.2.6 **Test**: User can view all members of their TauNet network
Description: The SRS specifies that the user will be able to display all of the users within their network if they would like to. This test verifies that ability.
Input: The user types view while the server is running.
Output: The list of TauNet users is displayed to them.
Result: PASS

2.3 TauNet Client:

The testing approach for this section is again focused mainly on acceptance testing of the TauNet requirements outlined in the SRS. The tests in this section also imply that this part of the program can be used with a keyboard/monitor. Desk checks were performed on all code that pertain to this section.

2.3.1 **Test**: Message is in the correct format. (Headers & Message)
Description: The TauNet protocol specifies the format for headers and line endings to be in.

This test verifies that they are in the correct format by messaging Bart's echo server and verifying no error message is received.
Input: The string 'This is a test' is sent as a message to the echo server.
Output: The sent message is relayed by the echo server back to the sending TauNet node and displayed on the screen. If any error is present it will be displayed in the received message.
Result: PASS – Initially I didn't pass this test because I wasn't including \r\n between the headers and the message body instead I was just using a /n for a blank line. After I fixed that one error I achieved the pass.

2.3.2 **Test**: Message length is no longer that 1KB in length.
Description: The TauNet protocol specifies that the max message size to be sent is 1KB. With 90 bytes for the header, and 10 bytes for the IV this means that there is 1024 – 100 = 924 bytes of available message space for a user to type a message. This test verifies that max is not exceeded.
Input: A message of 925 bytes is entered.
Output: An error message is received and the user is told to enter a new message.
Result: PASS

2.3.3 **Test**: User is able to view all user's within their network.
Description: The SRS specifies that the user should be able to view all members of their network if they would like to. This test verifies that is a possibility.
Input: When at the main menu of the program the user enters a 2 and presses enter.
Output: The list of all TauNet nodes within the network are displayed to the user.
Result: PASS

2.3.4 **Test**: User is able to send messages to other users within their network.
Description: The SRS specifies that a user will be able to send 1-to-1 messages to another user of their choosing. This test verifies that capability.
Input: At the main menu the user enters a 1, enters the name of the receiving user, and enters a message and presses enter.
Output: The notification of 'Message Sent Successfully' is displayed if the message was sent.
Result: PASS – In addition to the successful message there is another one that notifies if the user was unavailable.

2.4 Additional Testing:

2.4.1 **Test**: TauNet Operates on Raspberry Pi 2 with Raspbian OS
Description: The SRS specifies that TauNet be able to operate on Raspberry Pi 2 and Raspbian OS. This test is actually confirmed by running all other tests on the Raspberry Pi 2.
Input: All tests previously mentioned.
Output: A passing result for all previously mentioned tests.
Result: PASS

In addition to the tests provided in the sections above I wanted to explicitly state that I tested with other users. Mainly with Eric Tsai. And I also tested a reasonable amount with Bart's echo server. Screen captures of these testing sessions are included in the screen shots folder.