# CHAPTER 6

# SYSTEM TESTING

## 6.1. Overview

Chapter 6 contains the basic methods of black and white box testing, unit testing, integration testing and system testing. There are five modules tested based on modules created in system implementation including two phases of testing to test the reliability of Agent Communication Protocol and Algorithm and the effectiveness of Agent Verification Protocol and Algorithm as mentioned in research methodology in Chapter 1. Three tests have been done to examine the proposed solutions as mentioned in Chapter 2.

## 6.2. Testing Approaches

To test SAAIDS, five test modules have been created based on system modules from system components as well as in implementation phase. The modules include all modules in the implementation phase except for modules in Detection component because their implementation is using Snort as tool for information filter, data analysis and response. Therefore, there are five test modules tested which are Administrator, Agent Communication, Agent Verification and Response.

## 6.3. Coding Testing

In coding testing, there are two approaches used which are white box and black box testing. Black box and white box testing methods were used to test and measure the:

a. Completeness (Black-box testing and White-box testing)
b. Correctness (Black-box testing and White-box testing)
c. Reliability (White-box testing)
d. Maintainability (White-box testing)

### 6.3.1. White Box Testing

The white box testing is used to test specific paths through the code and is performed to reveal problems with the internal structure of a program. White box analysis involves analyzing and understanding every path through a program for their directions or chosen paths. White box testing is typically very effective in finding programming errors and implementation errors in software. A fundamental strength that all white box testing strategies share is that the entire software implementation is taken into account during testing, which facilitates error detection even when software specification is vague or incomplete. There are two types of white box testing used which are base line testing and cycle testing.

Base line testing is done on logical path of programs. Figure 5.1 to 5.4 shown the tested programs based on test modules which have been figured into path graph procedures. Cycle testing is done on several cycles of program's path. This testing is shown in figure 5.1 to 5.4 along with base line testing.

#### 6.3.1.1 Administrator Module

**Procedure: Administrator**

L1 : get loginID

L2 : if loginID <> administrator.name

L3 : go to L1 or exit

L4 : else get password and hash()

L5 : if hash(password) <> administrator.hash(password)

L6 : go to L1 or exit

L7 : else load administrator interface

L8a : case1 : view log files

L8b : case2 : control agent

L9 : end

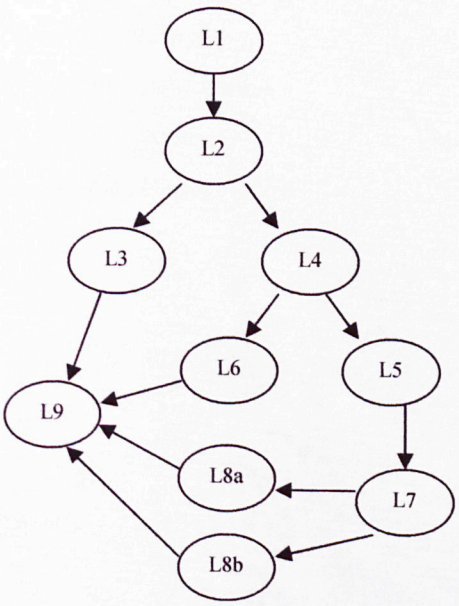


**Figure 6.1: Path Graph for Login Procedure**

Syclomatic complexion calculation is based on V(G) = R + 1 which R represents area rounded by nodes and arrows. For path graph for Administrator module procedure, the syclomatic complexion = 3 + 1 = 4.

### 6.3.1.2 Agent Communication Module

a. Receive Message Procedure

**Procedure: Receive Message**

L1 : decrypt(msg) = (R, $m_i$, $d_i$, $mt_i$, $p_i$)
L2 : get $p_i$ and count $m_i$
L3 : if $m_i > 1$ then
L4 : if $p_i = 1$ then casep1
L5 : else if $p_i = 2$ then casep2
L6 : else if $p_i = 3$ then casep3
L7 : else do $m_i$
L8 : casep1: if $m_i$ = $mdersp_i$ then alert($mdersp_i$)
L9 : else if $m_i$ = $mversp_i$ then alert($mversp_i$)
L10: casep2: if $m_i$ = $mdet_i$ then detect($mdet_i$)
L11: else if $m_i$ = $mver_i$ then verify($mver_i$)
L12: casep3: if $m_i$ = $mderst_i$ then log($mderst_i$)
L13: else if $m_i$ = $mverst_i$ then log($mverst_i$)
L14: end

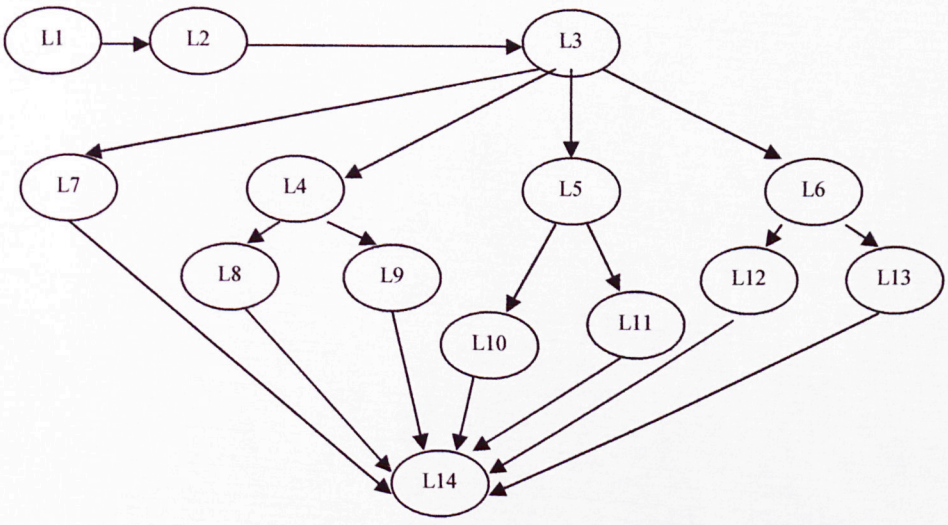


**Figure 6.2: Path Graph for Receive Message Procedure**

For path graph for Receive Message procedure in Agent Communication module, the syclomatic complexion = 6 + 1 = 7. This procedure is applied on Send Message procedure because both procedures have similar complexion of paths.

b. Encrypt and Decrypt Procedure

**Procedure: Encrypt**

L1 : get B, μ and Pb

L2 : if Pb = true then

L4 : computes Va, Za

create msg

computes Ca = msg*Za mod μ

return Va, Ca

L5 : else return failure Pb = false

**Procedure: Decrypt**

L6 : if Va and Ca = true then

L7 : msg = $Ca*Va^{-Sb}$ mod μ

create msg = ($R, m_i, d_i, mt_i, p_i$)

L7 : else return failure Va and Ca = false

L8 : end

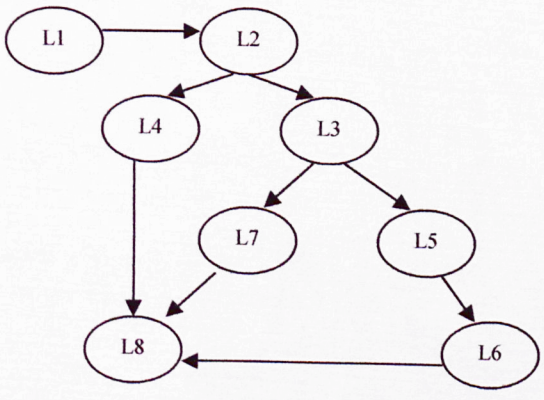


**Figure 6.3: Path Graph for Encrypt and Decrypt Procedure**

For path graph for Encrypt and Decrypt procedure in Agent Communication module, the syclomatic complexion = 2 + 1 = 3.

### 6.3.1.3 Agent Verification Module

**Procedure: Verifier**

L1 : get B and μ

L2 : if $t = t_S$ then

L3 : computes Pb, Vb

create msg = ($R$, $m_R$, $d_S$, $t_S$)

computes hi = H(msg[i] || Vb)

computes sign = Rb + hi * Sb mod μ - 1

return Pb, Vb, msg, sign

L4 : else return failure t = false

L10: if $t <= t_R$

L11: if($mveri_R$ == true) then

L12: log($R$, $d_R$, $t_R$)

$mveri_S$ = ($i$, $d_S$, $t_S$)

L13: else verifiresp()

L14: else verifiresp()

L15: end

**Procedure: Verified**

L5 : If(Pb, Vb, msg, sign = true) then

L6 : computes hi' = H'(msg[i] || Vb)

create msg = ($S$, $m_R$, $d_S$, $t_S$)

L7 : verify (sign)

if($\beta^{sign}$ mod μ = Vb $Pb^{hi'}$ mod μ) then

L8 : return $mveri_R$

L9 : else return failure Pb, Vb, msg, sign == false

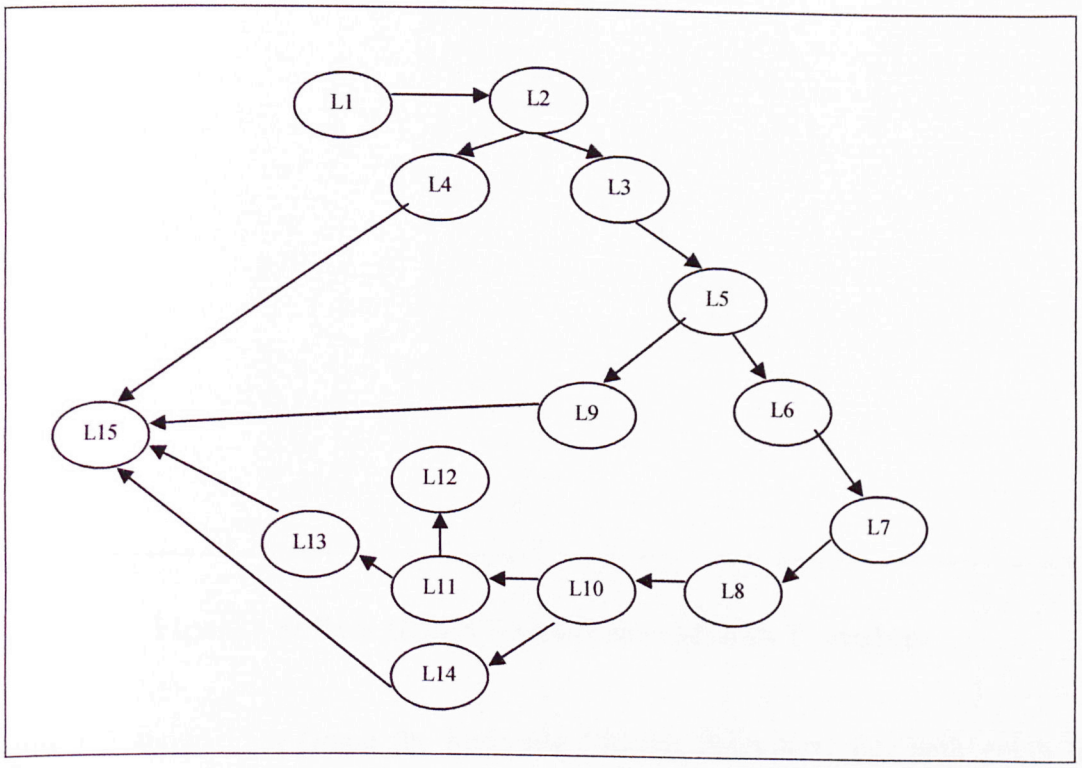


**Figure 6.4: Path Graph for Verifier and Verified Procedure**

For path graph for Encrypt and Decrypt procedure in Agent Communication module, the syclomatic complexion = 3 + 1 = 4.

### 6.3.1.4 Response Module

```
Procedure : Response
L1   : get mrsp
L2a  : if mrsp = detectresp(mdersp)
         then sndmesg(mdersp)
L3   : exit
L2b : else if mrsp = verifiresp(mversp)
         then sndmesg(mversp)
L4   : exit
E    : end
```

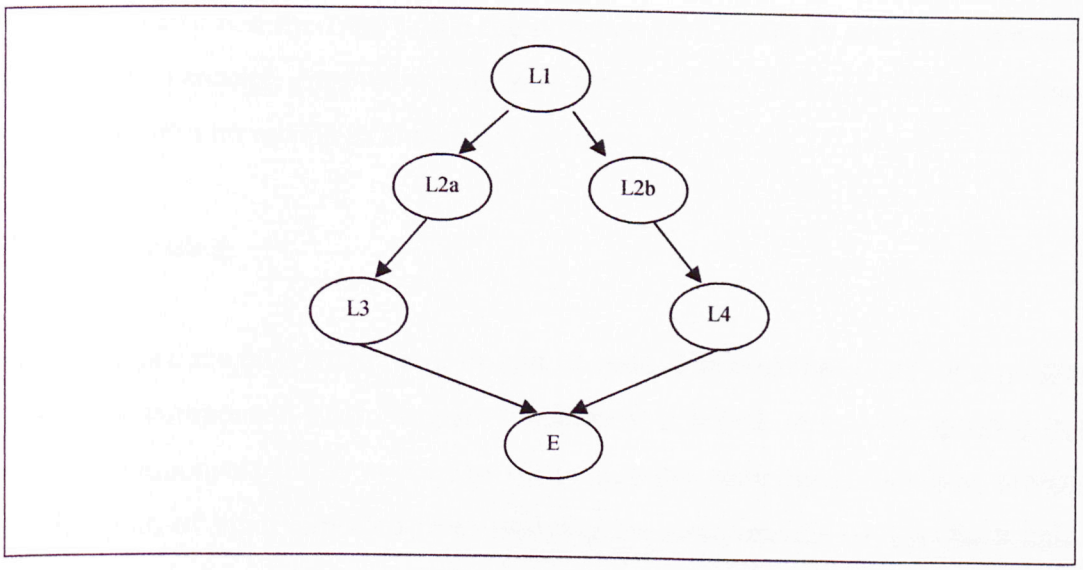


**Figure 6.5: Path Graph for Response Module Procedure**

Figure 6.5 shows Path Graph for Response Module Procedure. For path graph for Administrator module procedure, the syclomatic complexion = 1 + 1 = 2.

### 6.3.2. Black Box Testing

Black box testing is performed to access how well a program meets its requirements, looking for missing or incorrect functionality. Black box tests treat the program like a black box; what happens in the program is invisible and unimportant to the user. Black box test cases only look at specific inputs and outputs of an application. Black box analysis refers to analyzing a running program by probing it with various inputs. For SAAIDS, basically for test modules tested in black box testing is based on the requirements of the system. The black box testing technique solved the following errors:

a. Requirements are not fulfilled or mistaken errors
b. Interface errors
c. Implementation errors in terms of effectiveness and performance
d. Errors in beginning and ending of programs

This test is performed by overloading the program with inputs to see where it reaches its maximum capacity, especially with real time systems. This black box testing is done along with unit testing as shown in next section.

## 6.4. Unit Testing

Unit testing is done on a small piece or unit of code. The most basic unit of a program or system is component. Each program component is tested on its own, isolated from the other components in the system to verify that the component functions properly with the types of input expected from studying the component's design. Each unit is tested based on the module. For this system, the unit testing is tested based on form due to most of the module already included in each form. The evaluate result for each test cases which are tested based on module is summarized in a table test summary. In the test summary, details of submodule, testing technique used or activity, result and action are summarized in a table. The 'pass' condition under nit testing fulfilled the expected outputs.

### 6.4.1. Administrator Module

Unit testing on Administrator module is done through Administrator interface provided for system administrator. The module involves three submodules which are Login, View Log and Control Agent. Login submodule must do authentication process on loginID and password to enter the interface. Then, View Log submodule must view log based on administrator chosen log. Meanwhile, Control Agent submodule must launch control on agent based on administrator chosen control as shown in Table 6.1.

**Table 6.1: Unit Testing for Administrator Module**

| No. | Submodule | Activity | Result | Action |
|---|---|---|---|---|
| 1 | Login | Enter loginID and password.<br>• If corrected password entered, directed to Administrator interface.<br>• If not, login or password error notification interface viewed and administrator needs to reenter the correct loginID and password or exit. | Pass | |
| 2 | View log | Choose log.<br>• If detection log, detection log viewed.<br>• If verification log, verification log viewed. | Pass | |
| 3 | Control agent | Choose control<br>• If start, agent started.<br>• If pause, agent paused<br>• If resume, agent resumed.<br>• If stop, agent stopped. | Pass | |

### 6.4.2. Agent Communication Module

Unit testing on Agent Communication module is done within a controlled network environment which consists five hosts with each of them installed with SAAIDS. The agent communication tested between agents in the hosts as shown in Table 6.2.

**Table 6.2: Unit Testing for Agent Communication Module**

| No. | Submodule | Activity | Result | Action |
|---|---|---|---|---|
| 1 | Receive Message | • Decrypt received message<br>• Set priority if message count more than one<br>• Launch message process based on priority using priority cases and message options | Pass | |
| 2 | Send Message | • Encrypt message to be sent<br>• Set priority if message count more than one<br>• Send message process based on priority using priority cases and message options | Pass | |
| 3 | Encrypt | • Get Pb and test if Pb is true or else return failure Pb is false<br>• If true, computes Va and Za, create message, computes Ca<br>• Return Va and Ca. | Pass | |
| 4 | Decrypt | • Get Va and Ca and test if Va and Ca are true or else return failure Va and Ca are false<br>• If true, create message | Pass | |

### 6.4.3. Agent Verification Module

Unit testing on Agent Verification module is done within a controlled network environment which consists five hosts with each of them installed with SAAIDS. The agent verification tested between agents in the hosts as shown in Table 6.3.

**Table 6.3: Unit Testing for Agent Verification Module**

| No. | Submodule | Activity | Result | Action |
|---|---|---|---|---|
| 1 | Verifier | Begin<br>• Get t and test if t is similar to $t_S$ or else return failure t is false<br>• If true, computer Pb and Vb, create msg, computes hi and sign<br>• Return Pb, Vb, msg and sign<br>Receive<br>• Get t and test if t is quicker or similer to $t_S$ or else launch verifiresp()<br>• If true, test if $mveri_R$ is true or else launch verifiresp()<br>• If true, launch log(R, $d_R$ , $t_R$) and $mveri_S$ = (i, $d_S$ , $t_S$) | Pass | |
| 2 | Verified | • Get Pb, Vb, msg and sign and test them if are true or else return failure Pb, Vb, msg and sign are false<br>• If true, computes hi', create msg and verify sign<br>• If verified, return $mveri_R$ | Pass | |

### 6.4.4. Response Module

Unit testing on Response module done after an alert produced after intrusion detected. The testing is done within a host which installed with SAAIDS as shown in Table 6.4.

**Table 6.4: Unit Testing for Agent Response Module**

| No. | Submodule | Activity | Result | Action |
|---|---|---|---|---|
| 1 | Response | Get response message<br>• If response message is detection response message, send detection response message<br>• If response message is verification response message, send verification response message | Pass | |

## 6.5. Integration Testing

Integration testing is the process of verifying whether the system components work according to the system requirement. The integration testing is carried out when all components involved in the module have been tested. The intention is to expose faults in the interaction between software modules and forms. All existing modules are combined into a working system.

For this system, the integration testing consists Administrator module which the only forms integrated with other software modules. The integration testing on Administrator module has been done along with unit testing before.

## 6.6. System Testing

After all the modules have been completed and properly tested through all types of testing, the entire system needs to be validated by conducting a system testing. System testing is a series of different tests designed to fully exercise the system to uncover its limitation and measure its capabilities. This system testing is carried out to verify the functional and nonfunctional requirement of the system, which is already explained in Chapter 3.

In system testing, the event list testing is carried out. In the event list, all possible triggers must be exercised and the researcher validates the actual result. The focus is on response time testing on Agent Communication and Agent Verification module. Response time here is defined as how long the recipient agent response to both modules process. The duration of both module processes have to commit the duration required as mentioned in system functional requirements for Transceiver in Chapter 3 to guarantee the successful completion of both protocols. System testing on both modules is done within a controlled network environment which consists five hosts with each of them installed with SAAIDS. Each agent installed is equipped with verification code H'(VID).

### 6.6.1. Testing on Agent Communication Module

#### 6.6.1.1 Event 1: Installing New Agent

Event 1 created to test the communication between agents after installing a new agent in the network. Following is the testing procedure for Event 1 on Agent Communication Module as follows:

a. Install and configure AgentS agent with verification code into the network.
b. Install and configure second Agent001 agent with verification code into the network.
c. Send the verification message for new agent installed to be responded.
d. Observe the verification message response flowing between both agents.
e. Observe the time response between agents.

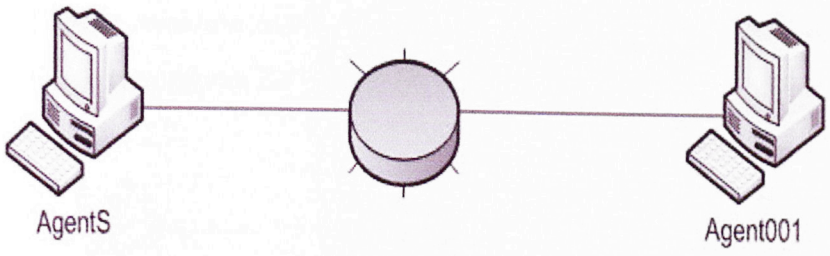


**Figure 6.6: Testing Event 1**

AgentS sends verification message for new agent installed to Agent001. Then, Agent001 will respond to the message by sending its verification code back to the AgentS to be verified as a new agent. Response time counted as soon as sender agent received acceptance notification from receiver agent. The required period for completion of agent communication module process is within 8 second. Figure 6.6 shows the testing environment for Testing Event 1.

### 6.6.2. Testing on Agent Verification Module

#### 6.6.2.1 Event 2: Without the Presence of Fake or Unauthorized Agent

Event 2 created to test the communication between agents without the presence of fake or unauthorized agent. Following is the testing procedure for Event 2 on Agent Verification Module as follows:

a. Install and configure first agent AgentS with verification code into the network.
b. Install and configure second Agent001 and third Agent002 with verification code into the network.
c. Observe the verification message response flowing between both agents.
d. Observe the time response between agents.

AgentS sends verification message to Agent001 and Agent002. Then, Agent001 and Agent002 will respond to the message by sending its verification code back to the AgentS to be verified as an existing agent. Response time counted as soon as verifier

agent received acceptance notification from verified agent. The required period for completion of agent communication module process is within 10 second. Figure 6.7 shows the testing environment for Testing Event 2.

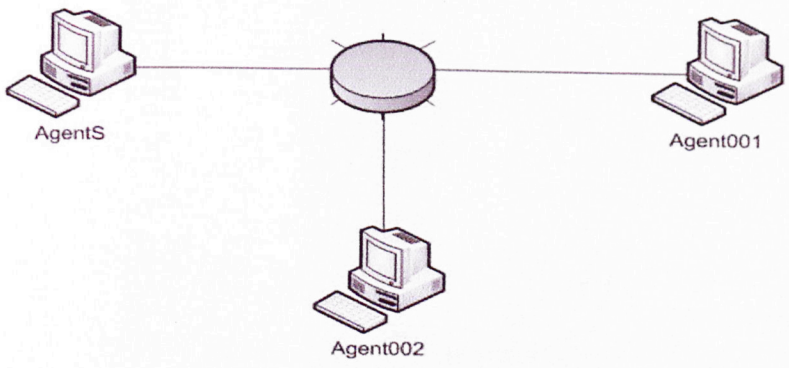


**Figure 6.7: Testing Event 2**

#### 6.6.2.2 Event 3: With the Presence of Fake or Unauthorized Agent

Event 3 created to test the communication between agents with the presence of fake or unauthorized agent. Following is the testing procedure for Event 3 on Agent Verification Module as follows:

a. Install and configure first AgentS and second Agent001 into the network.
b. Install and configure an AgentX without configuring the verification code into the network.
c. Observe the verification message response flowing between both agents.
d. Observe the time response between agents.

AgentS sends verification message to Agent001 and AgentX. Then, Agent001 and AgentX will respond to the message by sending its verification code back to the AgentS to be verified. Response time counted as soon as verifier agent received acceptance notification from verified agent. The required period for completion of agent communication module process is within 10 second. Figure 6.8 shows the testing environment for Testing Event 3.

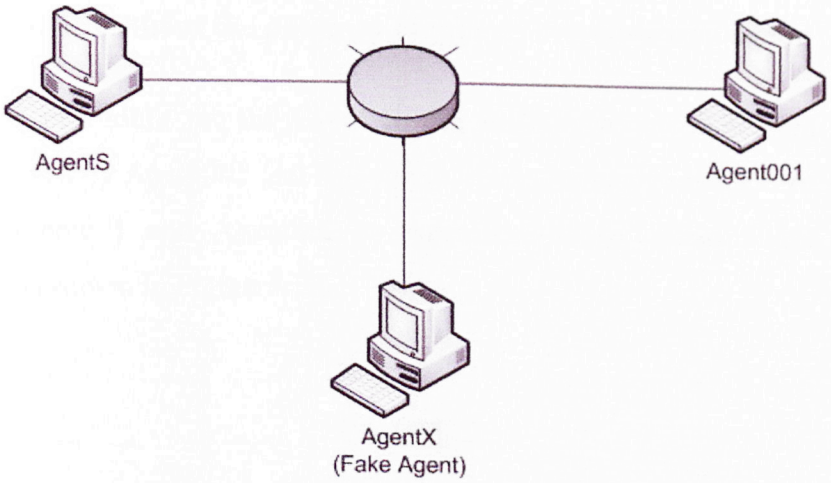


**Figure 6.8: Testing Event 3**

### 6.6.3. Testing Results

#### 6.6.3.1 Event 1: Installing New Agent

After ran the procedure for the Event 1, the verification message sent by AgentS securely received by Agent001 and the response for verification code for the new agent Agent001 to AgentS sent within 8 seconds as shown in Figure 6.9.

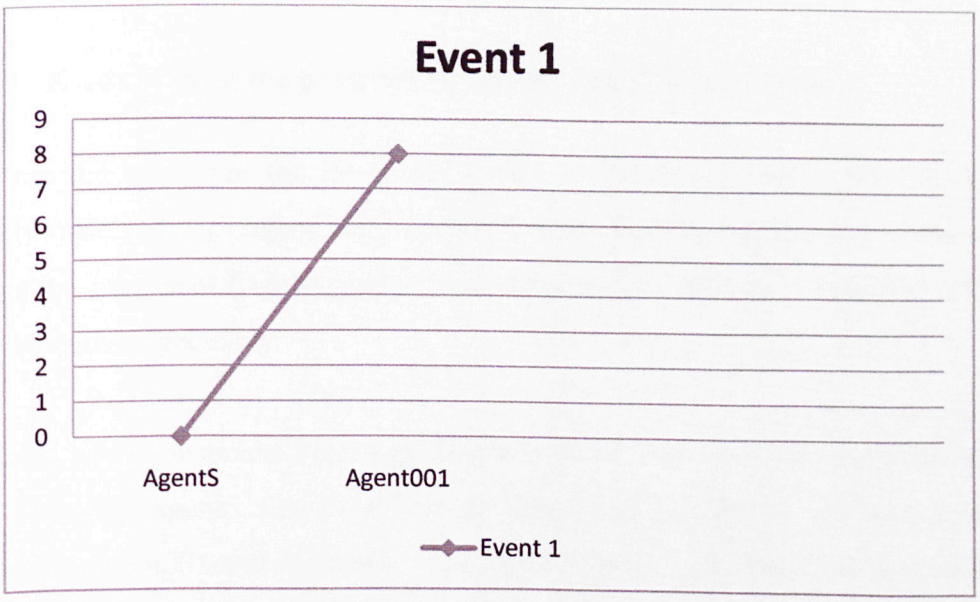


**Figure 6.9: Result for Testing Event 1**

### 6.6.3.2 Event 2: Without the presence of fake or unauthorized agent

After ran the procedure for the Event 2, the verification message sent by AgentS securely received by Agent001 and Agent002. Then, the response for verification code sent from Agent001 and Agent002 to AgentS is done within 5 and 6 seconds respectively as shown in Figure 6.10.

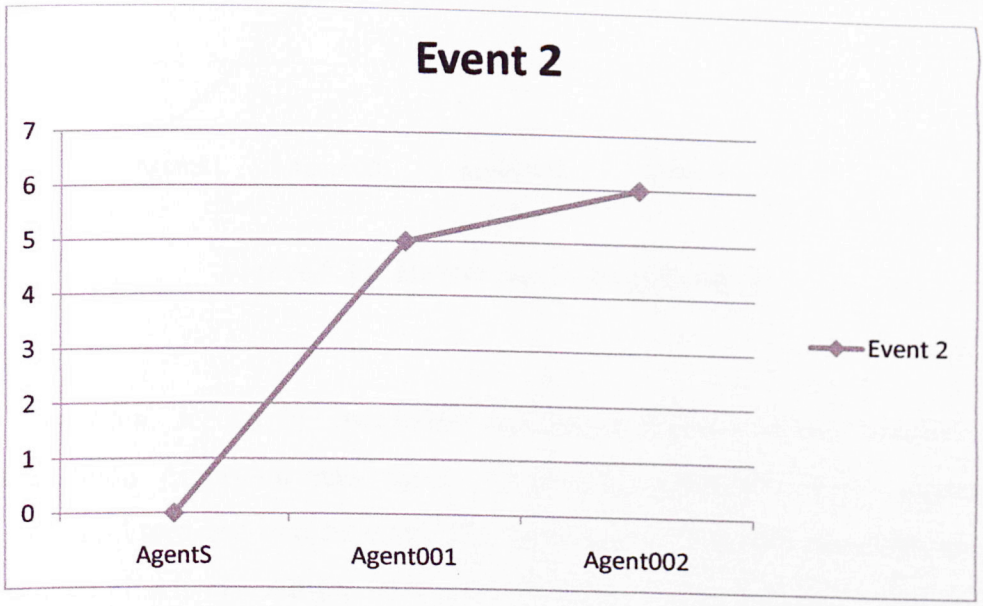


**Figure 6.10: Result for Testing Event 2**

### 6.6.3.3 Event 3: With the presence of fake or unauthorized agent

After ran the procedure for the Event 3, the verification message sent by AgentS securely received by Agent001, Agent002 and AgentX Then, the response for verification code sent from Agent001 and Agent002 to AgentS is received within 5 and 6 seconds respectively.

But, there was no response from AgentX which took more than 10 seconds in waiting time. Then, the AgentS views verification alert warning in the screen and send alert message to Agent001 and Agent002. As a result AgentS, Agent001 and Agent002 kill their connection to AgentX. The result is shown in Figure 6.11.

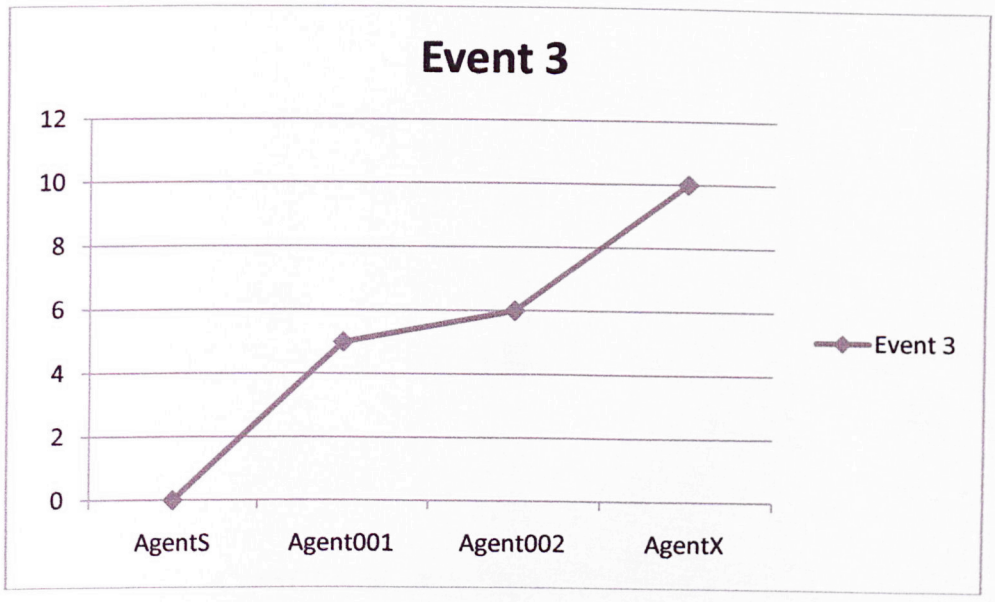


**Figure 6.11: Result for Testing Event 3**

From the result, it can be concluded that both modules which consist Agent Communication Algorithm and Agent Verification algorithm were successfully designed, developed and implemented. The graph shows that both protocols are valid and operated effectively. All the testing results have shown that this system is ready for acceptance testing but in this project no acceptance testing done on this system.