CHAPTER 5

SYSTEM IMPLEMENTATION

5.1. Overview

Chapter 5 discusses implementation of the system. In this chapter, coding for each module in this system and interfaces that related with the coding is being discussed in details.

5.2. System Module Design

The overall system can be viewed as five modules which are combinations of classes from Security, Transceiver, Detection and Monitor packages. The modules are Administrator, Agent Communication, Agent Verification, Detection and Response. All the modules are shown in Figure 5.1. Detection module implementation is not shown in this chapter because its processes used Snort to do intrusion detection process. Meanwhile, Response module implementation is included in Agent Communication module and Agent Verification module respectively. The inclusion intends to show the process flow of both modules implementation.

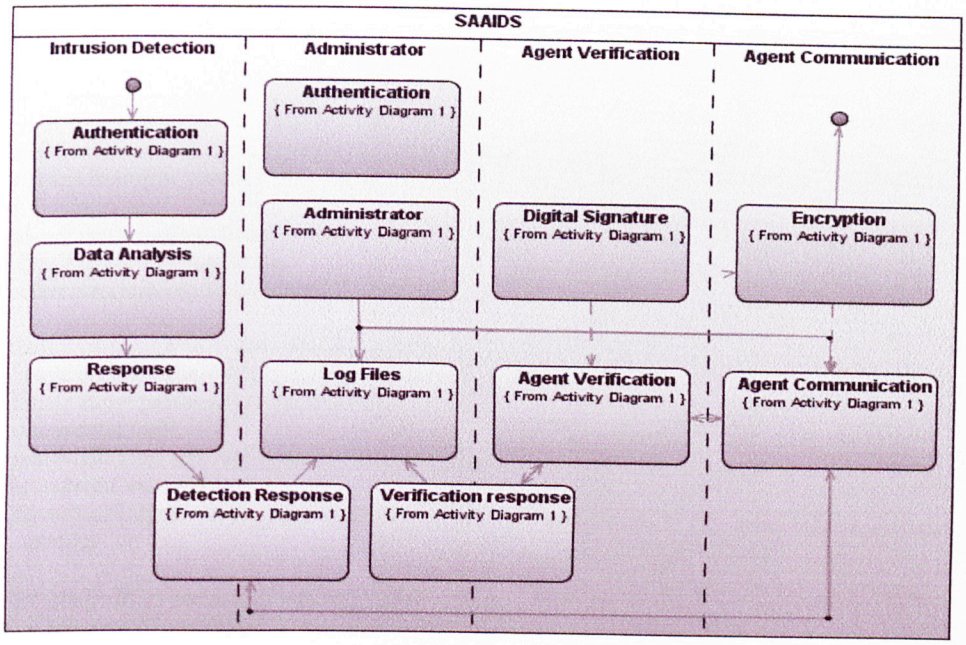

Figure 5.1: System Modules

5.3. Administrator Module

In this section, the structure of Administrator module is explained. It consists of the important part of coding, explanation of the coding and the interface that is related with this module. Administrator has Login sub module which use SHA-1 hash algorithm to encrypt administrator's password as shown in Figure 5.2.

5.3.1. Coding

```
private void authenticate(){
    if(LoginPreferences.forModule(AdministratormanagementPanel.class).get("user",
"").equals(this.form.getUsername())){
      try {
        char[] passwordFromForm = null;
        char[] passwordFromPref =
LoginPreferences.forModule(AdministratormanagementPanel.class).get("pass", "").toCharArray();

        String passwordPref = new String(this.form.getPassword());
        MessageDigest sha = MessageDigest.getInstance("SHA-1");
        byte[] tmp = passwordPref.getBytes();
        sha.update(tmp);
        passwordFromForm = byteArrToString(sha.digest()).toCharArray();
        int correctCount = 0;
        if(passwordFromForm.length != passwordFromPref.length){
          exit();
        }
        for (int i = 0; i < passwordFromPref.length; i++) {
          if (passwordFromPref[i] == passwordFromForm[i]) {
            correctCount++; }
        }
```

Figure 5.2: Administrator Modules Coding (continues)

```
        if (passwordFromPref.length == correctCount) {
          //do nothing here
        } else {
          exit();}
      } catch (NoSuchAlgorithmException ex) {
        Exceptions.printStackTrace(ex);
      }
    } else errlogin(){
      exit();
    } }

private static String byteArrToString(byte[] b){
  String res = null;
  StringBuffer sb = new StringBuffer(b.length * 2);
  for (int i = 0; i < b.length; i++){
    int j = b[i] & 0xff;
    if (j < 16) {
      sb.append('0');
    }
    sb.append(Integer.toHexString(j));
  }
  res = sb.toString();
  return res.toUpperCase();
}
```

Figure 5.2: Administrator Modules Coding (continued)

5.3.2. Coding Explanation

In Administrator module, loginID and password are retrieved from login form. Then, the loginID and password used by function authenticate() to be compared with stored loginID and password in AgentDB. In the process of login into Administrator interface, when loginID are matched, function authenticate() used SHA-1 to encrypt input administrator password and password from database to be compared. The comparison of password used length of char array which converted from byte and string format. To convert from byte to string format, IntegertoHexString() function used.

Meanwhile, LoginID comparison is done in text string format. If neither LoginID nor password are not matched or null, error login message interface will be viewed. Figure 4.3 shows Login interface and Figure 4.4 shows Error Login Message interface.

5.3.3. Interfaces

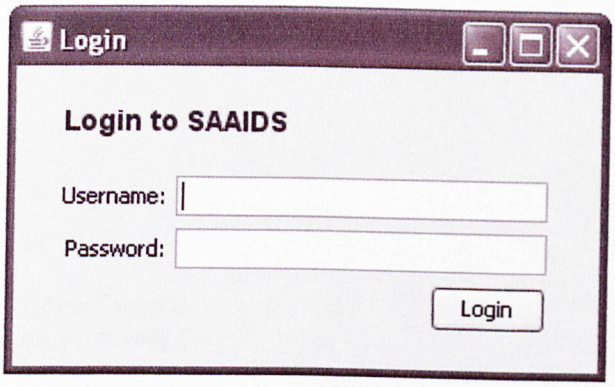

Figure 5.3: Login Interface

Figure 5.3 and 5.4 show Login Interface and Error Login Message Interface.

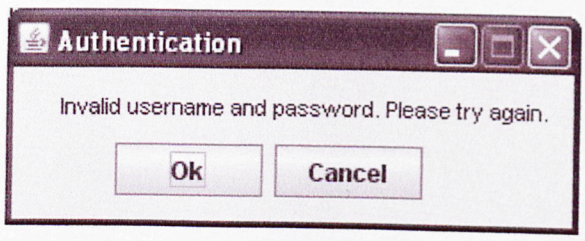

Figure 5.4: Error Login Message Interface

5.4. Agent Communication Module

In this section, the structure of Agent Communication module is explained. It consists of the important part of coding, explanation of the coding and the interface that is related with this module as shown in Figure 5.5.

5.4.1. Coding

```
Generates B, μ

enc(Pb)
{
    If (Pb == true)
    {
        for(i = 0 to i + 1)
        {
            Va = β^Ra mod μ
            Za = Pb^Ra mod μ
            msg[i] = (R, m_i, d_i, mt_i, p_i)
            Ca = msg[i]
            Ca = msg*Za mod μ
            i++
            return Va, Ca
        }return failure(i > 0)
    }return failure(Pb == false)
}

dec(Va, Ca)
{
    If(Va, Ca == true)
    {
        for(i = 0 to i + 1)
        {
            msg[i} = Ca*Va^-Sb mod μ
            (R, m_i, d_i, mt_i, p_i) = msg[i]
            i++
            return success (msg[i] == true)
        }return failure(i > 0)
    }return failure(Va, Ca == false)
```

Figure 5.5: Agent Communication Module Coding

5.4.2. Coding Explanation

The implementation of Agent Communication module is based on Agent Communication Algorithm which has been discussed in Chapter 3. The implementation process begins with alert produced by Detection module which is stored in Detection Log in Detection database (DetectionDB) for detection alert and Verification Log in Agent database (AgentDB) for verification alert. Response sub modules in Detection Module doing searches for new alert for intrusion detection in the database and launch detection response process. In addition to that, Response sub module broadcast the detection alert message to all agents in the system through Agent Communication module.

5.4.3. Interface

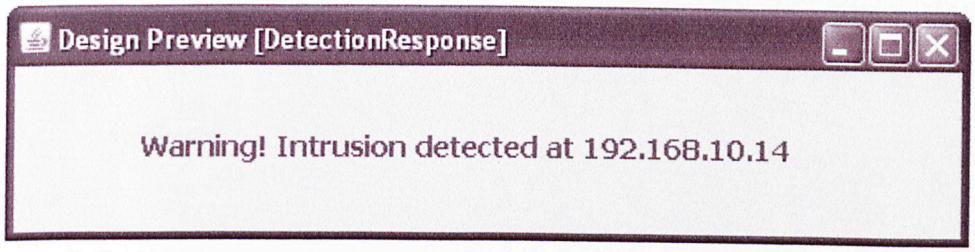

Figure 5.6: Detection Response Module Interface

There is an interface for detection warning alert broadcasted throughout the system as shown in Figure 5.6.

5.5. Agent Verification Module

In this section, the structure of Agent Verification module is explained. It consists of the important part of coding, explanation of the coding and the interface that is related with this module as shown in Figure 5.7.

5.5.1. Coding

```
Generates B, μ

verifier()
{
    If (t == tS)
    {
        for(i = 0 to i + 1)
        {
            Computes Pb ← β^Sb mod μ
            Computes Vb ← β^Rb mod μ
            Create msg[i] ← (R, mR, dS , tS)
            Computes hi ← H(msg[i] || Vb)
            Computes sign ← Rb + hi * Sb mod μ - 1
            i++
            return Pb, Vb, msg, sign
        } return failure(i > 0)
    } return failure(t == false)

    if(t <= tR)
    {
        if(mveriR == true)
        {
            log(R, dR , tR)
            mveriS = (i, dS , tS)
        } else
        {
            verifiresp()
        }
    }else
    {
        verifiresp()
    }
}

verified(Pb, Vb, msg, sign)
{
    If(Pb, Vb, msg, sign == true)
    {
        for(i = 0 to i + 1)
        {
            Computes hi' ← H'(msg[i] || Vb)
            Create msg[i] ← (S, mR, dS , tS)
            verify (sign)
            if(β^sign mod μ == Vb Pb^hi' mod μ)
            {
                return mveriR
            }
        }return failure(i > 0)
    }return failure(Pb, Vb, msg, sign == false)
}
```

Figure 5.7: Agent Verification Module Coding

5.5.2. Coding Explanation

The implementation of Agent Verification module is based on Agent Verification Algorithm which has been discussed in Chapter 3. The implementation process begins with alert produced when authorized agent detected and verification response information stored in Verification Log in Agent database (AgentDB). Then, Verification Response sub module broadcast the verification alert message to all agents in the system through Agent Communication module.

5.5.3. Interface

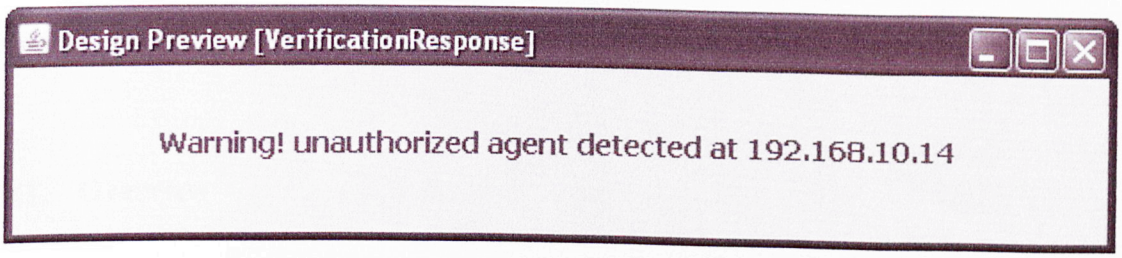

Figure 5.8: Verification Response Module Interface

There is an interface for verification alert warning broadcasted throughout the system as shown in Figure 5.8.