

MIDP Security

9.1. Tujuan:

Setelah selesai praktikum ini, kita dapat:

- Membuat sertifikat pada program
- Autentifikasi sebuah MIDlet
- Membuat Midlet yang dapat menangani hubungan statik.
- Membuat Midlet yang dapat menangani hubungan dinamik.

9.2. Latihan 1: Membuat sertifikat pada program

Pada latihan ini, kita dapat menyelesaikan langkah-langkah sbb:

- Menggunakan J2ME Wireless Toolkit untuk memberikan sertifikat pada program.
- Menambah kunci public pada file JAD dengan baik.
- Meberitahukan pada MIDlet-Certificate-n-m: bahwa sertifikat sudah ditambah pada file JAD.

Tujuan dari latihan ini dapat menggunakan J2ME Wireless Toolkit untuk memberikan sertifikat pada program. Sekarang game StarShip Battle sudah selesai, kita harus memprotek aplikasi. Dengan menambah sertifikat pada aplikasi, kita tidak hanya memprotek keaslian intelektual tapi juga memprotek pemakai dari download yang tidak diinginkan.

Persiapan:

Untuk menyiapkan latihan ini lakukan setting computer pada lokasi direktori: labdir = _____

Langkah 1— Menggunakan J2ME Wireless Toolkit untuk memberikan identitas pada program

Pada langkah ini kita menggunakan J2ME Wireless Toolkit untuk memberikan sertifikat pada program.

Copy program dari direktori ..\apps\Lab8_Exercise1\src ke direktori ..\apps\Lab9_Exercise1\src dan ikuti step ini:

1. Jalankan J2ME Wireless Toolkit dan buka proyek lab9_exercise1.
2. Dari menu J2ME Wireless Toolkit pilih Project dan pilih Sign.
3. Untuk membuat kunci baru pilih New Key Pair dan lakukan setting sbb:
 - Alias: StarShipBattleKey
 - Domain c:=localhost
 - O=:Sudent Productions
4. Pilih Create dan pilih trusted dari menu pull-down
5. Pilih OK

Sertifikat telah kita pasang dan kita telah membuat kunci baru yang dapat digunakan untuk menandai game StarShipBattle kita.

Langkah 2— Menambah kunci public pada file JAD dengan baik

Keamanan sebuah aplikasi dan API tergantung pada domain proteksi yang baik. Secara umum aplikasi terinstal dalam domain untrusted pada J2ME Wireless Toolkit. Ketika pada domain untrusted maka aplikasi harus ijin. Pada MIDP 2.0 memungkinkan untuk menggunakan sertifikat dengan baik. Pada langkah 1 kita memasang sertifikat secara manual. Kunci private otomatis diinstal pada emulator dan kunci public sekarang ada. Pada langkah ini tambahkan kunci public pada file JAD dengan baik.

Ikuti step berikut:

1. Jalankan J2ME Wireless Toolkit dan buka proyek lab9_exercise1.
2. Dari menu J2ME Wireless Toolkit pilih File dan pilih Utilities.
3. Pilih Sign Midlet
4. Pilih sertifikat StarShipBattle dari list disebelah kiri
5. Pilih Sign Midlet Suite
6. Dari file chooser, nafiikasi file Lab9_exercise1.jad yang terdapat pada direktori apps\lab9_Exercise1\bin

Langkah 3— Meberitahukan pada MIDlet-Certificate-n-m: bahwa sertifikat sudah ditambah pada file JAD

9.3. Latihan 2: Autentifikasi sebuah MIDlet

Pada latihan ini kita install aplikasi menggunakan Over-the-Air (OTA)

Tujuannya adalah pemakai terproteksi ketika menjalankan API tanpa ijin. Meskipun file JAD berisi sertifikat pemakai tetap pada kondisi prompt untuk ijin. Jika pemakai memilih fungsi help dari game sementara aplikasi dijalankan. Domain sertifikat dan proteksi hanya relevan dan digunakan ketika aplikasi diinstal menggunakan OTA. J2ME Wireless Toolkit mempunyai simulasi proses OTA dari aplikasi transfer dan install. Juga setelah aplikasi diinstal di emulator. Aplikasi berjalan pada domain proteksi sebenarnya pemakai tidak pada kondisi prompt untuk ijin untuk menjalankan proteksi API.

Persiapan:

Untuk menyiapkan latihan ini lakukan setting computer pada lokasi direktori: labdir = _____

Langkah — Menginstal Aplikasi Menggunakan OTA

Ikuti step berikut:

1. Jalankan J2ME Wireless Toolkit dan buka proyek lab9_exercise1.
2. Dari menu J2ME Wireless Toolkit pilih Project dan pilih Run via OTA.
Disini dipilih emulator dan pemakai berinteraksi dengan emulator Application Management Software (AMS)
3. Pada emulator pilih Apps
Emulator dapat melihat semua Midlet yang diinstal pada peralatan
4. Pilih select pada emulator
Direktori proyek Hypertext Markup Language (HTML) dapat terlihat
5. Pilih Menu dan pilih select
Halaman Lab9_Exercise1.html mulai download
6. Pilih install
AMS mulai download file JAD. Ketika file JAD didownload emulator menampilkan informasi Midlet
7. Pilih install lagi
AMS download file Java Archive (JAR) dan install aplikasi pada emulator. Setelah install lengkap kita lihat suite ditambahkan pada list aplikasi ketika simulator OTA dijalankan

8. Untuk verifikasi install berhasil jalankan aplikasi dengan memilih lab9_Exercise1 dari list kemudian pilih Menu dan pilih Launch
9. Setelah aplikasi dijalankan pilih Help dari menu utama. Kita tidak dapat di prompt untuk izin menjalankan hubungan jaringan

9.4. Latihan 3: Pendaftaran sebuah hubungan menggunakan konfigurasi

Midlet

Pada latihan ini kita mengembangkan game StarShipBattle untuk memberitahukan pemakai ketika score tertinggi baru terdaftar dengan server utama.

Keuntungan Push Register adalah informasi dapat dikirim ke peralatan tanpa pemakai membutuhkan. Informasi dikirim server dari email, sales update, sport information, untuk merubah score game. Bagaimanapun juga Push Register memperkenalkan ide khusus karena peralatan pemakai tidak ingin dihentikan oleh informasi yang tidak dibutuhkan.

Persiapan:

Untuk menyiapkan latihan ini lakukan setting computer pada lokasi direktori: labdir = _____

Langkah — Membuat Midlet baru

Copy program dari direktori ke apps\Lab9_Exercise1\src ke apps\Lab9_Exercise5\src Ikuti step berikut:

4. Deklarasikan sebuah class baru yaitu HighScoreNotification extends class Midlet didefinisikan dalam javax.microedition.midlet dan implementasikan interface CommandListener yang didefinisikan dalam paket javax.microedition.lcdui
5. Deklarasikan property dari class HighScoreNotification:
 - List score – Menampilkan score yang dikirim server ke peralatan
 - Display deviceDisplay – menampilkan score ke pemakai
 - Command exit – pemakai keluar dari Midlet
6. Deklarasikan method dari class HighScoreNotification:
 - void startApp() – populasi property class oleh instant tipe data
Setelah lengkap method getConnection() dapat dipanggil
 - void pauseApp() – tidak mempunyai implementasi
 - void destroyApp(Boolean unc) - tidak mempunyai implementasi

- `public void commandAction(Command c, Displayable d)` – Menutup aplikasi
 - `void getConnection()` – Merubah list hubungan dari Push Registry
tiap hubungan sebuah thread baru dibuat yang menangani jaringan untuk merubah informasi.
7. Mendeklarasikan inner class yaitu `ConnectionHandler` yang mengimplementasikan interface `Runnable`
 8. Mendeklarasikan property `String connURL` dalam interface `ConnectionHandler`. Properti merepresentasikan URL untuk menghubungkan tanggung jawab thread. Properti ini set instant dan memanggil method konstruktor.
 9. Mendeklarasikan method dalam interface `ConnectonHandler`:
 - `ConnectionHandler(String cURL)` – Konstruktor digunakan untuk set property `connURL`
 - `void run()` – berisi informasi pengiriman jaringan dan mengupdate score.
 10. Set property J2Me Wireless Toolkit:

```

MIDlet-Permissions:
javax.microedition.io.Connector.http, javax.microedition
.io.PushRegistry, javax.microedition.io.Connector.socket
, javax.microedition.io.Connector.serversocket
MIDlet-Push-1: socket://:5000, HighScoreNotification, *

```

9.5. Latihan 4: Pendaftaran sebuah hubungan menggunakan pemrograman

Pada latihan ini kita mendaftarkan sebuah hubungan static menggunakan JAD. Pada latihan ini kita merubah aplikasi yang dikembangkan dalam latihan terakhir untuk menggunakan dinamik Push Registry dibandingkan pendaftaran static

Persiapan:

Untuk menyiapkan latihan ini lakukan setting computer pada lokasi direktori: labdir = _____

Langkah — Modifikasi Aplikasi `HighScoreNotification`

1. Copy program dari direktori ke `apps\Lab9_Exercise5\src` ke `apps\Lab9_Exercise6\src`
2. Tambahkan properti pada midlet `HighScoreNatification`:
 - `String url = "socket://:1024;`
 - `String data=null;`

3. Tambahkan method pada midlet HighScoreNatifcation:
 - boolean connectionRegistered(String URL) – Iterasi melalui semua hubungan yang didaftarkan dengan Push Registry. Jika URL ditemukan maka true dan jika tidak false.
 - void registerConn(String URL) – Daftarkan URL yang dilewatkan dengan Push Registry.
4. Setelah startApp() cek property URL yang dilewatkan Push Registry dengan memanggil method connectionRegistered(String URL). Jika false maka method registerConn(String URL) dijalankan lewatkan parameter property url. Iterasi menggunakan semua hubungan untuk menentukan jika ada data yang datang dan penting untuk ditangani.

9.6. Penyelesaian Latihan

Gunakan penyelesaian untuk mengecek jawaban anda pada latihan dalam lab.

Latihan 1: Membuat Sertifikat

Gunakan program 9-1

Latihan 2: Autentifikasi sebuah Midlet

Setelah aplikasi dijalankan kita dapat memilih Help dari menu utama tanpa prompt untuk ijin untuk menjalankan hubungan jaringan.

Latihan 3: Pendaftaran sebuah hubungan menggunakan konfigurasi Midlet

Bandingkan jawaban anda dengan contoh program 9-2 pada file HighScoreNotification.java.

Program 9-2 adalah contoh file HighScoreNotification.java – Static Registration

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
public class HighScoreNotification extends MIDlet implements CommandListener{
    private List scores;
    private Display deviceDisplay;
    private Command exit;
    public void startApp(){
        deviceDisplay = Display.getDisplay(this);
        scores = new List("Latest High Scores", Choice.IMPLICIT );
        exit = new Command("Exit",Command.EXIT, 1);
        scores.addCommand(exit) ;
    }
}
```

```

        scores. setcommandListener(this);
        getConnections();
        deviceDisplay.setCurrent(scores);
    )
    public void pauseApp(){
    }
    public void destroyApp(boolean unconditional){
    }
    public void commandAction(Command a, Displayable d){
        destroyApp(false);
        notifyDestroyed();
    }
    private void getConnections(){
        String [] connlist = PushRegistry.ListConnections(true) ;
        if(connList.length == 0){
            errorAlert();
        }else{
            for(int index = 0;index<connList.length;index++) {
                Thread handler = new Thread(new ConnectionHandler(connList[index]));
                handler.start();
            }
        }
    }

    private void erroralert ( ) {
        Alert.error = new Alert("Scores Unavailable", "I will notify you
        when\n the scores have\n changed", null, AlertType.INFO);
        error. setTimeout (Alert.FOREVER) ;
        deviceDisplay.setCurrent(error,scores);
    }
    class ConnectionHandler implements Runnable{
        private String connURL;
        public ConnectionHandler(String inURL) {
            connURL = inURL;
        }
        public void run(){
            try{
                ServerSocketConnection ssc =
                (ServerSocketConnection)Connector.open (connURL);
                SocketConnection sc = (SocketConnection)ssc.acceptAndOpen();
                InputStream in = sc.openInputStream();
                InputStreamReader inMsg = new InputStreamReader(in);
                char[] temp = new char[128];
                StringBuffer data = new String Buffer(512);
            }
        }
    }

```

```

        while(inMsg.read(temp,0,temp.length)!=-1){
            data.append(temp) ;
        }
        if(data.length()!=0){
            scores.append(data.toString().trim(),null) ;
        }
        inMsg.close();
        sc.close();
        ssc.close();
        System.out.println("Closed the connection");
    } catch(Exception e) {
        System.out.println("ERR: "+ connURL);
        e.printStackTrace();
    }
    System.out.println("Leaving run()");
}
}
}

```

Latihan 4: Pendaftaran sebuah hubungan menggunakan pemrograman

Bandingkan jawaban anda dengan contoh program 9-3 pada file HighScoreNotification.java.

Program 9-3 adalah contoh file HighScoreNotification.java – Dynamic Push Registration

```

import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.io.*;
public class HighScoreNotification extends MIDlet implements CommandListener{
    String url="socket://:5000";String data=null;
    TextBox msg;Command exit;
    public void startApp(){
        try{
            if (!connect.isConnected(url) ){
                registerConn(url);
            }
            String [] conns = PushRegistry.listConnections(true) ;
            for(int index = 0,index<conns.length;index++){
                data = handleConnection(conns [index] ) ;
            }
        } catch(Exception e) {
            System.out.println("ERR: instartApp(),') ;
        }
    }
}

```

```

        if(data==null) {
            data = "No Messages";
        }
        msg = new TextBox("Incoming Message", data, data.length() +1,
        TextFieldId.ANY) ;
        exit = new Command("Exit", Command.EXIT,1);
        msg.addCommand(exit);
        msg.setCommandListener(this) ;
        Display d = Display.getDisplay(this) ;
        d.setCurrent (msg);
    }
    public void pauseApp(){
    }
    public void destroyApp(boolean unc) {
    }
    public String handleConnection(String url) {
        String response = null;
        try{
            ServerSocketConnection ssc =
            (ServerSocketConnection)Connector.open (url ) ;
            Socketconnection sc = (SocketConnection) ssc. acceptAndOpen ( ) ;
            InputStream in = sc.openInputStream();
            InputstreamReader input = new InputstreamReader(in) ;
            char[] temp = new char[128];
            StringBuffer data = new StringBuffer(512);
            while(input.read(temp,0, temp.length)!=-1) {
                data.append(temp);
            }
            if(data.length()!=0){
                response = data.toString();
            }
            input.close();
        }catch(Exception e) {
            System.out.println("Err:handleConnection()");
        }
        return response;
    }
    public boolean connectionRegistered(String url) {
        boolean response = false;
        try{
            String [] regConns = PushRegistry.listConnections(false);
            for(int index=0; index < regconns.length; index++) {
                if (regConns [index] .equals (ur1)){
                    response = true;
                }catch(Exception e) {

```

```
        System.out.println("ERR: connectionRegistered()");
    }
    return responsei;
}
public void registerConn(String url) {
    String name = this.getClass().getName(); String filter="*";
    try{
        PushRegistry.registerConnection (url, name, filter);
    }catch(Exception e) {
        System.out.println("ERR: Connection");
    }
}
public void commandAction(Command c, Displayable d){
    destroyApp(false);
    notifyDestroyed();
}
}
```