|  |
| --- |
| // There are five different sleep modes in order of power saving: |

|  |
| --- |
| // SLEEP\_MODE\_IDLE - the lowest power saving mode |

|  |
| --- |
| // SLEEP\_MODE\_ADC |

|  |
| --- |
| // SLEEP\_MODE\_PWR\_SAVE |

|  |
| --- |
| // SLEEP\_MODE\_STANDBY |

**// SLEEP\_MODE\_PWR\_DOWN - the highest power saving mode**

**/usr/local/android-studio/bin$ ./studio.sh**

**Wimea called 30-06-2018**

**okenyronie@gmail.com**

**Android Studio Tutorial Beginners**

**HagoPlus Charles IT**

**Wamp server passwor ==”password”**

**make TARGET=avr-rss2 wimea-ict- rss2.hex**

**avrdude -c stk500v2 -P /dev/ttyUSB0 -p m256rfr2 -b 38400 -e -U flash:w:wimea-ict- rss2.hex Access Control**

All applications must communicate with other applications and operating system through the use of APIs when granting acess to API calls the following guidelines apply.

1. Use principle of least privilege to grant acess. Failure to follow the principle of least privilege can result in untrusted, unprivileged code performing unintended privileged operations. Privileged code can access privileged resources on behalf of an unprivileged caller by using the AccessController.doPrivileged() method.

2. Minimize privileged code

ensure that privileged code contains only those operations that require increased privileges.

Any other code contained within a privileged block must operate with the privileges of that block, increasing the attack surface.

3. Do not expose methods that use reduced-security checks to untrusted code.

Most methods that do provide security manager checks verify that every class and method in the call stack is authorized before they proceed. This prevents a sensitive method from acting on behalf of a malicious method that hides behind trusted methods in the call stack.

**Authentication and Password Management**

Guideline 1: Authentication information should not be logged.

Information such as user-names and passwords, is highly sensitive. This information should not be kept for longer than necessary nor where it may be seen, even by administrators. For instance, it should not be sent to log files and its presence should not be detectable through searches.

Guideline 2: Store passwords using hash functions.

Programs that store passwords as cleartext risk exposure of those passwords in a variety of ways. To limit the exposure of passwords is the use of hash functions such as SHA-256. MessageDigest msgDigest = MessageDigest.getInstance("SHA-256");

// Encode the password

byte[] hashVal = msgDigest.digest(password);

**Block malicious file uploads**

Java applications that accept file uploads must ensure that an attacker cannot upload or transfer malicious files. It is important to checking file size, content type, and file contents, among other metadata attributes before allowing it to be uploaded. The checkMetaData() method must be called before allowing File object passed to the method be uploaded.