```
/**
    * 静默安装APK（需要root权限）
    * @param apkFile APK文件
    * @return 是否安装成功
    */
   private fun installApkSilently(apkFile: File): Boolean {
       return try {
           LogManager.i("[$TAG] 开始静默安装APK: ${apkFile.absolutePath}")

           // 方法1：直接使用pm install命令（如果设备已经root）
           val success1 = tryDirectPmInstall(apkFile)
           if (success1) {
               LogManager.i("[$TAG] 直接pm install安装成功")
               return true
           }

           // 方法2：使用su命令
           val success2 = trySuPmInstall(apkFile)
           if (success2) {
               LogManager.i("[$TAG] su pm install安装成功")
               return true
           }

           LogManager.w("[$TAG] 所有静默安装方式都失败")
           false

       } catch (e: Exception) {
           LogManager.e("[$TAG] 静默安装异常: ${e.message}", e)
           false
       }
   }

   /**
    * 尝试直接使用pm install命令
    */
   private fun tryDirectPmInstall(apkFile: File): Boolean {
       return try {
           val command = "pm install -r \"${apkFile.absolutePath}\""
           LogManager.d("[$TAG] 执行直接安装命令: $command")

           val process = Runtime.getRuntime().exec(command)
           val exitCode = process.waitFor()

           val output = process.inputStream.bufferedReader().readText()
           val error = process.errorStream.bufferedReader().readText()

           LogManager.d("[$TAG] 直接安装输出: $output")
           if (error.isNotEmpty()) {
               LogManager.d("[$TAG] 直接安装错误: $error")
           }

           exitCode == 0 && output.contains("Success")

       } catch (e: Exception) {
           LogManager.d("[$TAG] 直接pm install失败: ${e.message}")
```

```
            false
        }
    }

    /**
     * 尝试使用su权限执行pm install
     */
    private fun trySuPmInstall(apkFile: File): Boolean {
        return try {
            val command = "pm install -r \"${apkFile.absolutePath}\""
            LogManager.d("[$TAG] 执行su安装命令: $command")

            // 使用ProcessBuilder来更好地控制进程
            val processBuilder = ProcessBuilder("su")
            val process = processBuilder.start()

            // 通过stdin传递命令
            val outputStream = process.outputStream
            outputStream.write("$command\n".toByteArray())
            outputStream.write("exit\n".toByteArray())
            outputStream.flush()
            outputStream.close()

            val exitCode = process.waitFor()

            val output = process.inputStream.bufferedReader().readText()
            val error = process.errorStream.bufferedReader().readText()

            LogManager.d("[$TAG] su安装输出: $output")
            if (error.isNotEmpty()) {
                LogManager.d("[$TAG] su安装错误: $error")
            }

            if (exitCode == 0 && output.contains("Success")) {
                true
            } else {
                LogManager.w("[$TAG] su安装失败，退出码: $exitCode")
                false
            }

        } catch (e: Exception) {
            LogManager.d("[$TAG] su pm install失败: ${e.message}")
            false
        }
    }
```