Investigating Android Scanning Procedures

The Goals:

1. Find out how Android calls the 802.11 scanning functions.
2. Find out how Android decide when to scan periodically (active scan).

Analyzing the Source Code

To analyze the scanning procedure, we will first take a look at how a single scan is performed, followed by background scanning.

The WiFi Scanner holds most of the information we are interested in. It contains information about which channels we are scanning, the callback once the scan is completed, has the entry point for starting single scans as well as background scans. One thing to note is that much of the background scan functionality has been depreciated:

```
 * @deprecated Background scan support has always been hardware vendor dependent. This supp
 * may not be present on newer devices. Use {@link #startScan(ScanSettings, ScanListener)}
 * instead for single scans.
```

So background scans have seemingly been replaced with asynchronous single scans. This also suggests that background scans prior to this change were indeed uniquely identified by the network interface vendor.

The source code for starting a scan is: `java /** * starts a single scan and reports results asynchronously * @param settings specifies various parameters for the scan; for more information look at * {@link ScanSettings} * @param executor the Executor on which to run the callback. * @param listener specifies the object to report events to. This object is also treated as a * key for this scan, and must also be specified to cancel the scan. Multiple * scans should also not share this object. * @param workSource WorkSource to blame for power usage * @hide */ @RequiresPermission(android.Manifest.permission.LOCATION_HARDWARE) public void startScan(ScanSettings settings, @Nullable @CallbackExecutor Executor executor, ScanListener listener, WorkSource workSource) { Objects.requireNonNull(listener, "listener cannot be null"); int key = addListener(listener, executor); if (key == INVALID_KEY) return; validateChannel(); Bundle scanParams = new Bundle(); scanParams.putParcelable(SCAN_PARAMS_SCAN_SETTINGS_KEY, settings); scanParams.putParcelable(SCAN_PARAMS_WORK_SOURCE_KEY, workSource); scanParams.putString(REQUEST_PACKAGE_NAME_KEY, mContext.getOpPackageName()); scanParams.putString(REQUEST_FEATURE_ID_KEY, mContext.getAttributionTag()); mAsyncChannel.sendMessage(CMD_START_SINGLE_SCAN, 0, key, scanParams); }`