ORIGINAL ARTICLE



Detecting GNSS Jamming and Spoofing on Android Devices

Nicholas Spens | Dong-Kyeong Lee | Filip Nedelkov | Dennis Akos

Colorado Center for Astrodynamics Research, University of Colorado Boulder

Correspondence

Nicholas Spens, Ann and H.J. Smead Aerospace Engineering Sciences, University of Colorado Boulder, USA. Email: Nicholas.Spens@colorado.edu

Abstract

Global navigation satellite system (GNSS) location engines on Android devices provide location and navigation utility to billions of people worldwide. However, these location engines currently have very limited protection from threats to their position, navigation, and time (PNT) solutions. External sources of radio frequency interference (RFI) can render PNT information unusable. Even worse, false signals or spoofing can provide a false PNT solution to Android devices. To mitigate this, four detection methods were developed and evaluated using native location parameters within Android: Comparing the GNSS and network locations, checking the Android mock location flag, comparing the GNSS and Android system times, and observing the automatic gain control (AGC) and carrier-to-noise density (C/N_0) signal metrics. These methods provide a powerful means to significantly increase the robustness of the Android GNSS-based PNT solution and are implemented in the GNSSAlarm Android application to demonstrate real-time jamming and spoofing detection.

Keywords

Android, GNSS jamming, GNSS spoofing, robust navigation

1 | INTRODUCTION

The inability of Android devices to detect spoofing is a growing concern. Spoofing may be performed intentionally by the device user or by an external source. One malicious motivation behind location-based service self-spoofing is profitability. Drivers for rideshare platforms such as Uber and Lyft can use a spoofed locations to appear as if they are waiting at a "hot spot" before they are even there, securing the rider ahead of other drivers who use their real locations (Hill, 2021). Drivers can also use software to falsely report longer routes than were actually driven, inflating the cost of the ride they provide (Adegoke, 2017).

External spoofing attacks can be performed as well. Zeng et al. (2017) presented a method by which someone using a mobile navigation service could be subtly misled up to a kilometer away from their initial destination and introduced the malicious applications of such a tool. Currently, it is possible to spoof an Android device's location with only GPS signals, despite the availability of other constellations and even network access points to aid in positioning (Miralles et al., 2020a). Therefore, the need for protection from spoofing is rapidly growing.

NAVIGATION, 69(3). Licensed under CC-BY 4.0 © 2022 Institute of Navigation



The GNSSAlarm application is in the process of being developed by the RF & SatNav Laboratory at the Colorado Center for Astrodynamics Research (CCAR) to demonstrate and refine a variety of tests that can be performed in real time to detect jamming/spoofing on Android devices. This application is a cumulative effort, continuing upon the work by Miralles et al. (2020a) and other research from the GNSS laboratory, and is built with framework from the open source GNSSLogger application.

For comparison, another application attempting to provide real-time spoofing detection is TORGI (Tactical Observation of RF GNSS Interference), which focuses mainly on the use of automatic gain control (AGC) and carrier-to-noise density (C/N_0) for spoofing detection (a test that is also explored with GNSSAlarm). However, Miralles (2021) demonstrated that the TORGI application failed to indicate a threat when a device running it was subject to radio frequency interference (RFI) spoofing. This can be attributed to the app's failure to consider AGC limitations on Android and its narrow range of tests, both issues that are addressed by GNSSAlarm.

2 | LOCATION MEASUREMENTS ON ANDROID

Position data on Android is provided through the native LocationManager class on Android. Through this class, listeners can be set up to constantly provide an app with location data, raw GNSS measurements, and National Marine Electronics Association (NMEA) messages. Basic location fix data comes in a data set organized by the location class such as latitude, longitude, altitude, bearing, accuracy, provider (the source of the data), time, and more (Android Developers, n.d.a). This class is all that is needed for an app to provide location-based functionality. It can be used to detect some forms of location spoofing/jamming but lacks the information required to evaluate the GNSS signals, themselves.

Location fixes in Android can be provided by three sources: The GPS Location Provider (GLP, which uses all strong GNSS signals available), the Network Location Provider (NLP), and the Fused Location Provider (FLP). The FLP uses both GNSS and network signals to calculate a position and is the recommended location fix for Android. It should be noted that within the FLP, GNSS contributions, when available, are heavily weighted in the fix calculation (Miralles et al, 2020a).

For more in-depth GNSS signal analysis, the GnssMeasurement class can be used. Introduced in Android 7.0 (API 24), this class provides a data set containing metrics about an individual GNSS signal, including C/N_0 , AGC, satellite ID, constellation type, carrier frequency, and more (Android Developers, n.d.b). Raw GNSS measurements gathered this way can be powerful tools for detecting spoofing/jamming, which will be demonstrated in this paper.

There are some limitations to this approach, however. First, it cannot be used on all Android devices, as those on a version of Android below 7.0 have no way of accessing the measurements. Approximately a quarter of all Android users are currently using a version below Android 7.0 (Rahman, 2020). Although this number is declining, it is by no means insignificant. Second, even among phones that do provide raw measurements, capability is varied. The ability to report measurements such as AGC and accumulated delta range (ADR) is dependent on the chipset used by the phone to process GNSS signals. Third, various vendors of Android GNSS chipsets have not fully standardized their measurements; an example is how AGC is reported differently across devices (this is further explored in Section 3.4.1).

The progression of relevant GNSS measurements on Android is summarized in Figure 1. The components used by the GNSSAlarm application are highlighted.

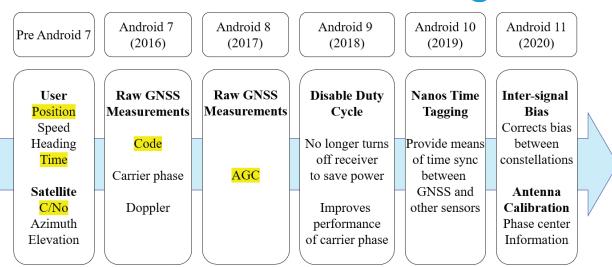


FIGURE 1 Development of GNSS measurements on Android over time

As can be seen in the figure, however, Android is a dynamic environment when it comes to the possible measurements, and it is possible that the application could use new measurements in the future.

For older phones that do not support GNSS measurements, some information can still be parsed out of National Marine Electronics Association (NMEA) message strings, most notably the $\mathrm{C/N_0}$. NMEA is a standard used by most commercial GNSS receivers to provide position, velocity, and time (PVT) information to users and is the main alternative to raw GNSS measurements for signal information on Android. However, there currently is no way to get AGC data without GNSS measurements. One of the co-authors of this paper, Lee, has submitted a proposal to add an AGC message to the NMEA standards, which could allow for an alternative method of retrieving AGC data on Android (Lee et al., 2020).

3 | DETECTION METHODS

The four detection methods will be summarized below, along with pertinent details on their implementation and the limitations of each. None of the methods have a particularly high computational cost and, as such, can all be run simultaneously for all GNSS-related data collected by the Android device.

3.1 | GNSS vs Network Location

The first detection method takes advantage of a powerful feature available to mobile phones: The ability to generate a location fix from both network signals and GNSS signals. A location is calculated independently by each location engine and the distance between the two is found. The test will alert when the discrepancy between the two location fixes is greater than a defined threshold.

The fused location engine used to provide the location fix for most Android applications currently prioritizes GNSS location information over the network location as demonstrated by Miralles et al. (2020a). Therefore, a spoofer needs only to replicate GNSS signals in order to mislead an Android device. By comparing



the GNSS-based location to the location from the NLP, GNSS targeted attacks can be detected.

The NLP position draws from two sources. The first is cell tower localization, which is relatively inaccurate, but widely available. However, the introduction of 5G towers has the potential to increase accuracy due to the high density of towers needed to support the network. The second source of NLP positioning data is Wi-Fi fingerprinting. Received signal strength (RSS) from nearby Wi-Fi access points is used in conjunction with a database of known access point locations to build a location estimate.

Ideally, both methods are used, however, in rural areas cell tower localization may be the only option, and in some areas there might not be enough cell towers or Wi-Fi points to even generate an NLP position. Further information about how the NLP is generated is provided by Nedelkov et al. (2020).

3.1.1 Network Location Model

To reliably use an NLP as a detection metric, the error in the NLP position must be accounted for. This error is dependent on two factors: The availability of Wi-Fi and cellular network access points and the NLP update rate (Nedelkov et al., 2020). Naturally, the fewer access points there are, the greater the potential for error in the position calculation. As for update rate, the NLP can only be updated every 5 seconds in Android. This means that if a user is traveling at a high speed, the error contributed by this factor has time to grow to a significant value in the time between fixes.

Prior work by Nedelkov et al. (2020) established the nominal accuracy of the NLP position in a variety of environments. Testing data was taken from two Android devices placed inside a vehicle driving in various environments. One device logged NLP data with the Wi-Fi enabled and the other did the same with Wi-Fi disabled.

Note the similarities in suburban and urban environments, as both have sufficient access points to generate a reasonably accurate solution. If Wi-Fi access points are not available or blocked by the user, the accuracy drops significantly, even more so for the less dense suburban environment.

3.1.2 | GNSS vs Network Location Test Implementation

To implement this test, a listener function was set up and location updates were requested at the maximum rate from both the GNSS and network location providers (every 1 s and 5 s, respectively). When a location update was received by the listener function, it was distributed to other classes through a custom interface. The spoofing detection class then received the updated fix and stored it, converting the most recent GNSS fix and the most recent network fix from geodetic coordinates to Earth-centered Earth-fixed (ECEF) coordinates. A distance could then be calculated and compared to the specified threshold.

For this test to be effective, the environment must be considered. From the tests used to generate Table 1, it was determined that if there were many Wi-Fi and cellular access points, this threshold could be set at or below 100 m without presenting many false flags. With an increased set of test data, better models could be established and the threshold could be determined dynamically in response to the available network access points.



TABLE 1 NLP Error in Various Environments

Environment	Mean Error (m) Wi-Fi & Cellular	Mean Error (m) Cellular Only
Suburban Navigation	43.86	348.3
Urban Navigation	40.93	193.4

Timing must also be considered. If the device is moving at a high speed, the test will only be valid at the time of the NLP update, as the true position and the GLP position (with its higher update rate) would diverge from the NLP position in-between fixes.

3.2 | Mock Location

The mock location test alerts when the device's current location fix is from a mock provider. A mock provider is a separate application on Android that overrides the standard GNSS engine and substitutes in user defined latitude/longitude/altitude location parameters to the GNSS location fix. This process is allowed on Android devices as long as it has been enabled in the Developer Options settings.

A wide variety of applications that generate mock locations are available on the Google Play Store and many also provide functionality to set routes that move the mock location over time. Location spoofing can, therefore, be performed without any hacking or modification of the Android platform.

In order to detect this, the boolean isFromMockProvider() can be used, which is a flag attached to every location fix provided by the location class. This incredibly easy to implement test provides a basic defense against software-based location attacks and should be the minimum layer of security for any location-dependent application. However, as will be shown later in Section 5.2, many applications do not appear to check for locations from mock providers as of December 2021.

3.2.1 | Mock Location Test Implementation

To implement this test, the same listener/interface setup was used as in Section 3.1.2. When a GNSS fix is received, the spoofing detection class checks the isFromMockProvider() flag, and alerts if it is true.

3.3 | GNSS vs System Time

The third test compares the time stamp from the received GNSS message to the system time. One method to simplify the complexity of GNSS spoofing is to perform a replay attack wherein signals are received at one location and sent at a later time to a re-radiator near the target device (Papadimitratos & Jovanovic, 2008).

Minimizing the time delay significantly increases the complexity required. This check of time synchronization is a methodology to increase the challenge for a potential spoofer. Similarly, spoofing attacks may target the time, as opposed to the position, reported by the receiver.

GNSS signals can be compared to the system clock within Android for consistency. Naturally, the system clock has some drift which limits how accurately the incoming time signals can be evaluated. However, Android devices can increase

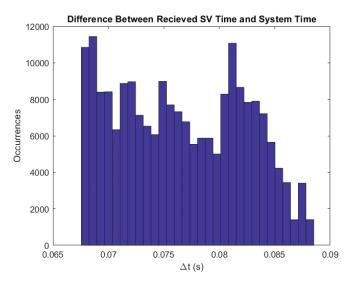


FIGURE 2 Difference between space vehicle (SV) time and system time under nominal conditions

accuracy by pulling time from the network (over cellular or Wi-Fi). Figure 2 shows a histogram of nominal time differences over a multi-hour test on the Samsung S20+ device.

3.3.1 | GNSS vs System Time Test Implementation

GNSS measurements are received through a callback function and, then, sent to the custom interface. The spoofing detection function must then convert the received SV time to the same time convention as the system clock before comparing the values (this conversion is different for all constellations). The largest time difference is kept and compared to the threshold. If the clock has recently been synchronized with the network, the error can be expected to be below 100 milliseconds. If the clock has not been synchronized recently, this error is harder to characterize, as there is variation between device clocks.

3.4 \perp AGC and C/N₀

The fourth test uses the GnssMeasurements class to examine incoming signal quality. Although there exist multiple methods using raw GNSS measurements to detect and mitigate potential RFI, the two measurements used in this application are AGC and C/N_0 . AGC provides an indication of how much power is coming into the receiver. When RFI is introduced, whether it is noise from jamming or signals from spoofing, the thermal noise floor is raised and AGC applies less gain (the measurement should decrease). C/N_0 also decreases in the presence of noisy interference, but a spoofing attack requires an artificial signal to overpower the real GNSS signal, so C/N_0 should remain the same or even increase in that scenario (Lee et al., 2021).

Manfredini et al. (2018) describe how the combination of these relationships enables receivers to distinguish between spoofing and jamming. If AGC decreases and C/N_0 decreases, jamming is likely. If AGC decreases and C/N_0 is relatively constant, spoofing is more likely. And if AGC is unchanged, then any form of

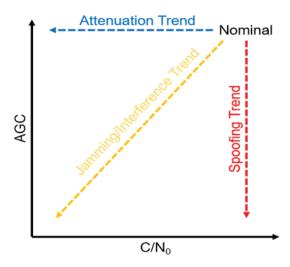


FIGURE 3 Expected AGC and C/N₀ trending

interference is unlikely and poor signal may be attributed to attenuation. Figure 3 shows how AGC can be plotted against C/N_0 to clearly visualize the expected trends.

To flag a threat to the user, a set of indicators is used, one for each constellation/band that receives distinct AGC and C/N_0 values. The indicators have two stages of alerts. If the AGC drops below a set threshold for interference and C/N_0 drops an equal amount or more, interference is likely and the respective indicators will turn yellow. If the same scenario happens but C/N_0 does not drop proportionally, the indicators will turn red to alert the spoofing threat. This test is powerful because it can isolate the source of an issue to a specific constellation or band, something that could potentially be used to screen out bad measurements and generate a safe solution in the future.

3.4.1 Obtaining a Useful AGC Measurement

AGC measurement implementation in Android is still somewhat immature. There are a number of inconsistencies and shortcomings in how measurements are reported across different hardware vendors, with three major issues at the moment.

First, the directional change of AGC under RFI is not consistent. For most phones, increased power results in a decrease in AGC but, for some (like the Huawei P30), an increase in power results in an increase in AGC.

Second, most chipsets report AGC on a relative scale, where the reported AGC is periodically re-scaled to a baseline value. This is demonstrated in Figure 4, first shown by Lee et al. (2021).

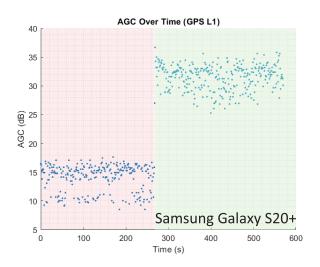
In the test, both phones started logging AGC data in the presence of RFI in the L1 band from a noisy notebook (this section of time is shown with a red background). Then, the RFI source was removed and the AGC predictably increased due to the decrease in observed power. However, after a short period of time, the Pixel 4 (and most other phones tested) returned to a value similar to that reported before (under RFI). This is an issue because if you were using AGC to detect spoofing or interference, you would not be able to see the impact after the AGC value was reset. Transient detection would be the only possible use.

Third, AGC is only available when GNSS measurements are received under the current implementation, despite it being a persistent measurement. If no



measurements are being received, AGC should be able to indicate if it is due to interference (low AGC) or simply attenuation (nominal AGC). This is demonstrated in Figure 5, where a ublox device with persistent AGC measurements provides a clear indication of the difference between the interference scenario and attenuation scenario (Lee et al., 2021). However, the Android device lacks AGC measurements in both scenarios so the distinction cannot be made. This issue could be fixed if the AGC is reported through a different class, which is possible due to the constant improvement of GNSS measurements on Android discussed in Section 2.

In addition to the improvement brought by hardware providers adhering to more consistent reporting standards, newer devices continue to provide a more sensitive, less noisy AGC measurement. In Figure 6, it can be seen that, in addition to the newer Pixel 6 (Broadcom BCM47765 GNSS chipset) reporting a greater mean AGC than the S20+ (Broadcom BCM47755), it also has a standard deviation of less than half of all the measurements collected in the same conditions. While the variation in AGC implementation is a limiting factor for this measurement and the test that depends on it, the current state is promising and will be powerful as the measurement improves.



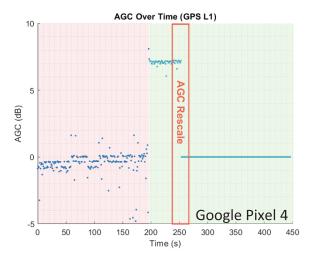
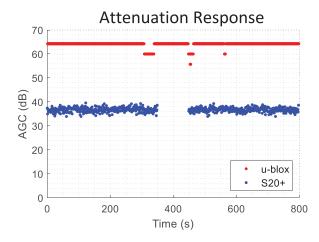


FIGURE 4 AGC reaction to removal of RFI



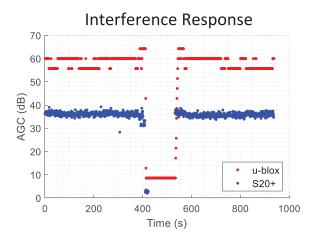


FIGURE 5 Android AGC response to attenuation and spoofing

SPENS ET AL.

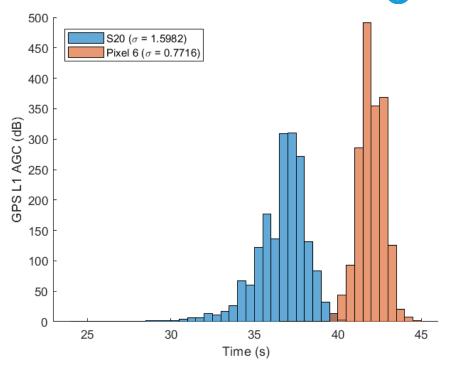


FIGURE 6 AGC values in nominal open-sky conditions

3.4.2 | AGC and C/N_0 Test Implementation

Raw GNSS measurements are received through the same mechanism described in Section 3.3.1. The set of measurements is iterated through to find the AGC of each constellation and the maximum $\mathrm{C/N_0}$ of a satellite in each constellation. These measurements are, then, scaled by pre-determined nominal values, which must be determined for each device model. The AGC is, then, checked against a threshold and the $\mathrm{C/N_0}$ relative to the AGC is also observed.

While current nominal AGC values and thresholds are device dependent, further standardization of the measurement may make these consistent across devices. The logic for this implementation is shown in Figure 7, where the AGC_THRESHOLD value sets the line that indicates RFI, while the CN0_THRESHOLD and CN0_SCALE values define the line that distinguishes between jamming and spoofing. CN0_SCALE determines the slope of the line (a slope of 1 is used unless testing can demonstrate otherwise) and CN0_THRESHOLD determines the offset. Points below/right of the line are considered abnormal for random interference.

Note that sometimes AGC is shared by constellations/bands with similar frequencies. For the S20+, GPS and Galileo share AGC for their L1 and L5 measurements. This means that in a given epoch, all GPS and GAL L1 measurements will report the same AGC value, which will falsely flag both constellations as problematic even when only one is being spoofed.

```
if((AGC[i]-nominalAGC[i])<AGC_THRESHOLD){
//Flag yellow (jamming)
if((CN0[i]-nominalCN0[i])>((AGC[i]-nominalAGC[i])*CN0_SCALE+CN0_THRESHOLD)){
    //Flag red (spoofing)
```



4 | THE GNSSALARM APPLICATION

The GNSSAlarm application applies all four metrics to incoming location data using the implementations described. Each metric was not applicable to every spoofing attack on its own, however, when combined, the scope of possible spoofing attacks was significantly narrowed. The app is still undergoing testing and development, but will be publicly available once the authors are comfortable with its performance.

4.1 | Application Structure

Location data from the location class, the GNSSMeasurement class, and NMEA messages are distributed through a custom interface (GNSSListener). Classes for logging (FileLogger) and UI/display (UILogger) implement GNSSListener and have access to the stream of data from the aforementioned sources. The FileLogger class records data of interest to a text file (when logging has been started), while the UILogger class displays additional metrics on the screen. Then, the SpoofingDetection class runs four different tests, the results of which are presented to allow the user to determine the likelihood of location manipulation as well as the type of attack. The app structure can be visualized in Figure 8.

4.2 | User Interface

The application's user interface (UI) is shown in Figure 9 with significant components labeled. Positioning information and a general spoofing/jamming indicator lie at the top with a map in the middle. Below it are the metrics used

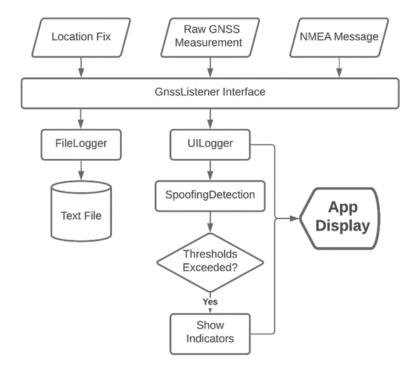


FIGURE 8 GNSSAlarm application structure



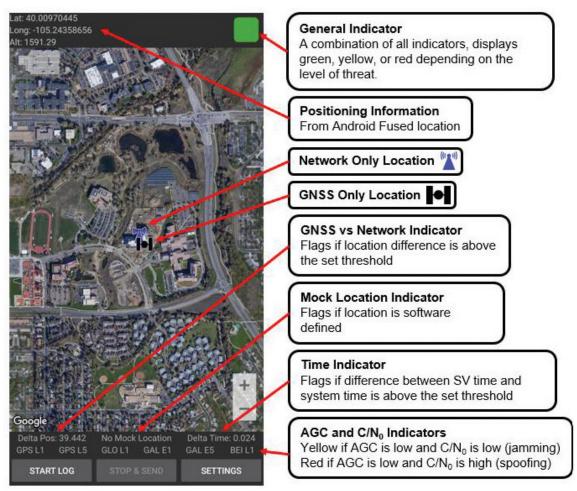


FIGURE 9 GNSSAlarm UI

to identify spoofing/jamming and controls for the logging functionality are at the bottom.

The general indicator shows the highest threat level observed by any test at the time. An informed user can then use the other provided metrics to determine the nature of the threat. All values are updated once every second, except for those dependent on network location, as network location may only be updated once every five seconds. The UI can be used to check the results of GNSS RFI alarm tests in real time.

When "START LOG" is selected, a text file and an SQLite database file are opened in the phone's storage. The data recorded here is easily customizable to meet the needs of different experiments. Currently, every GNSS measurement received at each epoch is recorded individually, and the time, constellation, frequency, satellite vehicle ID (SVID), C/N_0 , and AGC is saved. A satellite may appear twice per epoch if a signal is received in both the L1 and L5 bands. After the file is closed, a MATLAB script can be used to parse and post-process the data.

To adjust the thresholds used by the tests in the app, the "SETTINGS" button can be selected. An example configuration used on the S20+ is shown in Figure 10. Note that there are different $\mathrm{C/N_0}$ and AGC offsets for each constellation/band. These should be determined empirically for each device unless further standardization is



Nominal Values				
Constellation/Band	CN0 Offset (dB-Hz)	AGC Offset (dB)		
GPS L1	48.0	35.0		
GPS L5	26.0	30.0		
GLONASS L1	43.0	50.0		
GALILEO E1	48.0	35.0		
GALILEO E5	32.0	30.0		
BEIDOU L1	45.0	50.0		
Thresholds				
GNSS/Network Distance (m) 100.0				

Thresholds			
GNSS/Network Distance (m)	100.0		
GNSS/System Time (s)	.15		
AGC Threshold (dB)	5.0		
CN0 Threshold (dB-Hz above AGC)	5.0		
APPLY			

FIGURE 10 GNSSAlarm settings menu

implemented. As can be seen in the figure, the nominal ${\rm C/N_0}$ and AGC values can vary greatly even within one device.

4.3 | Application Scope

The indicators in the GNSSAlarm application are capable of identifying a wide range of attacks. Software location spoofing and time spoofing can be detected by the methods in Sections 3.2 and 3.3, respectively. Spoofing or jamming with a reasonable power advantage can be detected and classified by the AGC and C/N_0 test. Any location spoofing in an environment with plentiful network access points will be negated by the GNSS vs network location test.

There are still a few scenarios in which the application may not be able to detect a threat. The primary example is an environment with little to no network access



in conjunction with a matched power spoofing attack that is difficult to detect with coarse AGC measurements provided by current smartphones. There are signal quality monitoring methods that perform well under the matched power attacks that AGC struggles to identify, however, it is unlikely that the multiple correlator measurements for such analysis will be included in Android in the near future (Miralles et al., 2020b). However, matched power attacks are difficult to perform outside of controlled environments, and the AGC measurements provided by Android devices will continue to improve (as demonstrated in Figure 6).

5 | TESTING

The results of two testing scenarios are presented. These are used to validate the implementation of their respective tests in GNSSAlarm.

5.1 | Application Response to Interference

To test the reaction to interference, a Samsung S20+ running GNSSAlarm was brought close to a laptop that produces radio frequency emissions around the GPS L1 band. Shown in Figure 11, as the phone approached the laptop, the AGC for the L1 frequencies of GPS, GLONASS, Galileo, and BeiDou satellites dropped over 5 dB from nominal levels and C/N_0 dropped proportionally. Thus, jamming was detected and the respective indicators turned yellow.

The 5-dB threshold was set based on the range of AGC values observed during nominal testing on the S20+. This threshold will be different for all phones as shown with the improvement for the Pixel 6, so it can be easily changed in the app for the specific platform.

5.2 | Application Response to Software Spoofing

The screenshots in Figure 12 show the effect of a location spoofing application. On the left, the spoofing application is set to overwrite the user's real position (Boulder, CO) with a determined position (Albuquerque, NM). In the middle

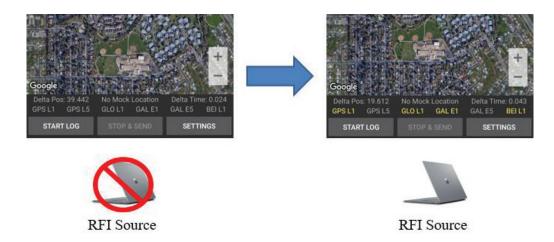


FIGURE 11 GNSSAlarm reaction to RFI jamming









FIGURE 12 GNSSAlarm reaction to software spoofing scenario

screenshot, it can be seen that the spoofing application has successfully replaced the user's location in the Google Maps application. On the right-hand screenshot, the GNSSAlarm application is shown (with the spoofer still active).

The GNSS and network position test has effectively identified the spoofer, and this can be seen in two ways. First, the "Delta Pos." value (center, left) is red, displaying a value that far exceeds the 100-m threshold set for this test. Second, a clear discrepancy can be seen on the map between the GNSS position (in black) and the network position (in blue, displaying the true position).

6 | CONCLUSION

The GNSSAlarm application is being developed with the goal of implementing effective tests for spoofing/jamming detection and providing a standalone application to visualize those test results in real time. The app takes advantage of the powerful GNSS insight added to Android by the GnssMeasurements class, as well as the more standard location information.

GNSSAlarm implements four tests based on prior work by Miralles and the RF & SatNav Laboratory. The app has validated its implementation by successfully detecting software-based mock location spoofing and hardware-based RFI jamming in real time, then communicating the threats effectively to the user. The proposed application is expected to provide Android device users and developers a robust tool for protection against potential GNSS RFI, facilitating a high accuracy and integrity location system for our society.

REFERENCES

Adegoke, Y. (2017). *Uber drivers in Lagos are using a fake GPS app to inflate rider fares.* Quartz Africa. https://qz.com/africa/1127853/

Android Developers. (n.d.a). *Location*. https://developer.android.com/reference/android/location/Location



- Android Developers. (n.d.b). GnssMeasurement. https://developer.android.com/reference/android/location/GnssMeasurement
- Hill, S. (2021). How Uber Drivers Use Fake GPS Apps for Quicker Ride Requests. iMyFone. https://www.imyfone.com/change-location/fake-gps-on-uber/
- Lee, D.-K., Miralles, D., Akos, D., Konovaltsev, A., Kurz, L., Lo, S., & Nedelkov, F. (2020). Detection of GNSS spoofing using NMEA messages. *2020 European Navigation Conference*, Dresden, Germany, 1–10. https://doi.org/10.23919/ENC48637.2020.9317470
- Lee, D.-K., Spens, N., Gattis, B. G., & Akos, D. M. (2021). AGC on Android devices for GNSS. *Proc.* of the 2021 International Technical Meeting of the Institute of Navigation, 33–41. https://doi.org/10.33012/2021.17823
- Manfredini, E., Akos, D. M., Chen, Y.-H., Lo, S., Walter, T., & Enge, P. (2018). Effective GPS spoofing detection utilizing metrics from commercial receivers. Proc. of the 2018 International Technical Meeting of the Institute of Navigation, Reston, VA, 672–689. https://doi.org/10.33012/2018.15595
- Miralles, D. (2021). *Reliable satellite radionavigation in the Android operating system* (Order No. 28416117) [Doctoral thesis, University of Colorado at Boulder]. ProQuest Dissertations & Theses Global.
- Miralles, D., Akos, D. M., Lee, D. -K., Konovaltsev, A., Kurz, L., & Lo, S. (2020a). Robust satellite navigation in the Android operating system using the Android raw GNSS measurements engine and location providers. 2020 European Navigation Conference, Dresden, Germany. https://doi. org/10.23919/ENC48637.2020.9317434
- Miralles, D., Bornot, A., Rouquette, P., Levigne, N., Akos, D. M., Chen, Y.-H., Lo, S., & Walter, T. (2020b). An assessment of GPS spoofing detection via radio power and signal quality monitoring for aviation safety operations. *IEEE Intelligent Transportation Systems Magazine*, 12(3), 136–146. https://doi.org/10.1109/MITS.2020.2994117
- Nedelkov, F., Lee, D. -K., Miralles, D., & Akos, D. M. (2020). Accuracy and performance of the network location provider in Android devices. Proc. of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation, 2152–2165. https://doi. org/10.33012/2020.17523
- Papadimitratos, P., & Jovanovic, A. (2008). GNSS-based positioning: Attacks and countermeasures. Milcom 2008 – 2008 IEEE Military Communications Conference, San Diego, CA. https://doi.org/10.1109/MILCOM.2008.4753512
- Rahman, M. (2020). *Android Version Distribution*. XDA Developers. https://www.xda-developers.com/android-version-distribution-statistics-android-studio/
- Zeng, K. C., Shu, Y., Liu, S., Dou, Y., & Yang, Y. (2017). A practical GPS location spoofing attack in road navigation scenario. *Proc. of the 18th International Workshop on Mobile Computing Systems and Applications*, 85–90. https://doi.org/10.1145/3032970.3032983

How to cite this article: Spens, N., Lee, D. -K., Nedelkov, F., & Akos, D. (2022). Detecting GNSS jamming and spoofing on Android devices. *NAVIGATION*, 69(3). https://doi.org/10.33012/navi.537