DEPARTMENT OF COMPUTER SCIENCE

UAV Resilience against GPS Spoofing Attacks

Analyzing the Effectiveness of the Direct Spoofing Technique against the NEO-7M GNSS Receiver

Badr Makhdoom

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Monday 9th September, 2024

Abstract

This dissertation explores the vulnerability of GPS receivers, particularly the U-blox NEO-7M used in commercial drones, to GPS spoofing attacks. The study evaluates the receiver's response under various spoofing scenarios across two operational states: acquisition and tracking. In addition, the effectiveness of the direct spoofing technique, a non-sophisticated spoofing method, is discussed. Using the Portapack H2, an add-on to the HackRF One SDR, experimental spoofing attempts against the receiver were conducted using static and dynamic spoofing signals that simulate different flight scenarios in order to observe the NEO-7M's response under controlled conditions. The tests revealed that the receiver is most susceptible during the initial acquisition state, where it has not yet locked onto the GPS satellite signals. In contrast, the tracking state demonstrated greater resistance to spoofing, but was affected by the spoofing signals nevertheless, mainly due to the transmission of high-power signals. The tests highlighted the critical window of attack to be during the signal acquisition state of the receiver, which opens room for spoofers to compromise GPS-dependant unmanned aerial vehicles (UAVs) even before launch. Potential defense mechanisms are then discussed as practical options for UAVs, and receivers specifically, to defend against spoofing attack techniques such as direct spoofing.

- I spent approximately 40 hours learning about the mechanism with which information, such as GPS satellite information, can be transmitted via radio frequency (RF) waves.
- I spent 150 hours learning about the global positioning system (GPS), how and why GPS spoofing attacks work, and the various methods of spoofing attacks possible.
- I analyzed the results observed from successfully spoofing the U-blox NEO-7M receiver with various multiple simulated paths and identified the states at which the receiver is most vulnerable.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Badr Makhdoom, Monday 9th September, 2024

Contents

1 Introduction **1**
1.1 Aims and Objectives . . . 1

2 Background **3**
2.1 Unmanned Aerial Vehicles (UAVs) . . . 3
2.2 GPS Technology in UAVs . . . 3
2.3 Software-Defined Radios . . . 5
2.4 GPS-SDR-SIM: GPS Signal Simulator . . . 7
2.5 Related Works . . . 8

3 Understanding GPS Spoofing Attacks **10**
3.1 GPS Receiver Mechanism . . . 10
3.2 Spoofed Signal Generation . . . 11
3.3 GPS Spoofing Techniques . . . 12
3.4 Attack Conditions: Receiver State, Attack Timing and Environmental Factors . . . 14
3.5 Defending against GPS Spoofing Attacks . . . 14
3.6 Spoofing Strategies to Evade Detection . . . 16

4 Methodology **18**
4.1 Portapack H2 HackRF One . . . 18
4.2 Portapack Tx Power Measurement . . . 20
4.3 GPS Antenna Used for Transmitting . . . 22
4.4 Ublox NEO-7M GPS Receiver . . . 23
4.5 PyGPSClient: Displaying Receiver Information . . . 23
4.6 Generating the Spoofing Signals . . . 24

5 Execution: Spoofing the NEO-7M **27**
5.1 Experiment Conditions . . . 27
5.2 Results . . . 27

6 Critical Evaluation **34**
6.1 Analysis of the Direct Spoofing Technique . . . 34
6.2 Defending Against Direct Spoofing Attacks . . . 37

7 Conclusion **38**
7.1 Future Research Directions . . . 38
7.2 Conclusion . . . 38

List of Figures

2.1 Trilateration and clock offset correction diagrams, taken from [44] 5
2.2 GPS-SDR-SIM script options 7

3.1 Process of generating spoofed GPS signals using a GPS signal simulator and transmitting the signals towards a target drone using an SDR. 12
3.2 Operation of a repeater in a synchronous GPS spoofing attack. Point A represents the reception of the authentic GPS signals. Point B represents the transmission of the spoofed signal. 13

4.1 A picture of the HackRF One SDR, taken from [1]. 18
4.2 Portapack H2 HackRF One 19
4.3 Screenshot showcasing Portapack H2 Mayhem firmware display. 19
4.4 HP 8594E Spectrum Analyzer used for power measurements. 20
4.5 The HP 8594E Spectrum Analyzer screen during base power level measurement. 21
4.6 The HP 8594E Spectrum Analyzer screen during gain power level measurement. 22
4.7 The Taoglas antenna used in demo execution. 22
4.8 Upper and Bottom sides of the U-blox NEO-7M GNSS receiver. 23
4.9 PyGPSClient graphical user interface with numbered sections. 24
4.10 List of BRDC files. The BRDC file used for demo execution is in the red box. 25
4.11 Spoofing 15 July Martyrs Bridge, Istanbul, Turkey. 25
4.12 Spoofing Cape Town, South Africa. 26
4.13 Spoofing Cape Town, South Africa. 26

5.1 Initial NEO-7M GPS locks to the spoofing signals of Istanbul, Havana, and Cape Town. . 28
5.2 Scatter plot displaying slight movement of the NEO-7M from the static spoofing signal. . 29
5.3 The circular path in Havana, displayed in the scatter plot. 29
5.4 Trace of the zigzag path in Cape Town, displayed in the scatter plot. 30
5.5 Satellite level views of the available satellites detected by NEO-7M during the transmission of the spoofing signals. 31
5.6 PyGPSClient display showing NEO-7M receiver information during transmission of the Havana dynamic spoofing signal. 33

6.1 Comparison between the ellipsoid and geoid models of Earth. Taken from [8]. 36

List of Tables

3.1 Start time measurements provided by Measurement Systems Ltd for an IKE GPS module [30] 11
3.2 Frequency Bands of GNSS Systems [35] 16

4.1 Setting options in GPSSim page of the Mayhem Firmware. 20
4.2 HP 8594E Spectrum Analyzer Settings 21

5.1 Summary of GPS Spoofing Results on NEO-7M Receiver in the acquisition state. 31

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Dr. Alma Oracevic.

Supporting Technologies

- **Portapack H2:** An add-on to the HackRF One SDR that transforms it into a standalone device. This was used for signal transmission.
- **Mayhem Firmware:** Installed on the Portapack H2 to enhances the device's capability. The GPSSim app within Mayhem was used directly for transmitting the spoofed GPS signals.
- **GPS-SDR-SIM:** A software tool for creating GPS signal files. This was used to simulate both static and dynamic GPS spoofing signals.
- **Taoglas GPS Antenna:** The antenna used to transmit the spoofed GPS signals within the GPS frequency range.
- **HP 8594E Spectrum Analyzer:** A tool used to measure and validate the components in the Portapack H2 that affect the transmission power of the GPS spoofing signals.
- **PyGPSClient:** A Python-based application that provides a graphical interface for monitoring U-blox GPS receivers on MacOS. It was used to display data from the NEO-7M receiver during the spoofing attempts.
- **U-blox NEO-7M GPS Receiver:** The target for the spoofing attacks.

Notation and Acronyms

GPS	:	Global Positioning System
GNSS	:	Global Navigation Satellite System
UAV	:	Unmanned Aerial Vehicle
SDR	:	Software-Defined Radio
ADC	:	Analog-to-Digital Converter
DAC	:	Digital-to-Analog Converter
SPS	:	Standard Positioning Service
PPS	:	Precise Positioning Service
IMU	:	Inertial Measurement Unit
AGC	:	Automatic Gain Control
PVT	:	Position, Velocity, and Timing
NMEA	:	National Marine Electronics Association
ECEF	:	Earth-Centered Earth-Fixed
LLH	:	Latitude Longitude Height
FM	:	Frequency Modulation
AM	:	Amplitude Modulation
FPGA	:	Field-Programmable Gate Array
RF	:	Radio Frequency
INS	:	Inertial Navigation System
LNA	:	Low-Noise Amplifier
PPP	:	Precise Point Positioning
AoA	:	Angle of Arrival
MEO	:	Medium Earth Orbit
SMA	:	SubMiniature version A
TX	:	Transmit
RX	:	Receive
I/Q	:	In-phase and Quadrature
GUI	:	Graphical User Interface
PRN	:	Pseudo-Random Number
TTFF	:	Time To First Fix
UBX	:	U-blox Binary Protocol
RTCM3	:	Radio Technical Commission for Maritime Services Type 3
NTRIP	:	Networked Transport of RTCM via Internet Protocol
SPARTN	:	State Space Representation Technique for GNSS Data
A-GNSS	:	Assisted Global Navigation Satellite System

Chapter 1

Introduction

Originally, unmanned aerial vehicles (UAVs) were invented and used for military use. Similar to other technologies like the internet, UAVs have transcended their initial military applications and are now found in numerous applications for civil and scientific use. Today, we find UAVs being introduced to areas such as film, precision agriculture, environmental monitoring, scientific research and the list goes on. At first, securing UAVs might not seem essential for some of these applications, like filmmaking, since an attack might seem unlikely. However, an attack threat is still possible, and that calls for the protection of UAVs. A compromised UAV is a safety hazard, because, put simply, a heavy object potentially falling out of the sky poses a risk to the people or facilities in its surroundings. In addition to such general case, there are critical UAV applications that cannot afford compromise, such as emergency response and border surveillance. In such instances, a UAV can be a determining factor in saving a person from a life-or-death situation or stopping drug trafficking across country borders.

Securing UAVs also enables innovation in sensitive areas and supports its integration into new products. Secure UAV systems will unlock new territories that are more welcoming for innovation and product development in previously uncharted areas: companies would be able to confidently integrate new technologies without compromising on users' security. Enhanced UAV security can popularize certain applications that were once thought of to belong to a technological utopia; aerial home delivery services, such as drone delivery, and autonomous urban air mobility (UAM) vehicles might become a reality. According to Straubinger et al. [41], recent advancements in engineering and technology had unlocked the means to build UAM vehicles.

In this dissertation, I delve into the mechanism behind GPS spoofing attacks, why GPS spoofing poses a critical threat to UAVs, and the defense mechanisms UAVs can employ to defend against these attacks. Through a practical experiment against the U-blox NEO-7M receiver[1], I analyze the effects of running a spoofing attack technique against the receiver. Then, I explore practical defense mechanisms that can be implemented on a drone to combat the attack. In this paper, I use the word "drone" interchangeably with "UAV" and "GPS spoofing attack" or "spoofing attack" with "spoofing". Note that only GPS signals that are intended for civilian use are considered in this paper.

1.1 Aims and Objectives

This project focuses on exploring the vulnerability of GPS receivers used in commercial drones, the U-blox NEO-7M in particular, to spoofing attacks and evaluates the effectiveness of the spoofing technique and existing defense mechanisms. The aims and objectives include:

- **Assessing the Vulnerability of the NEO-7M:** Assess the susceptibility of the NEO-7M receiver to GPS spoofing in 2 different operational states (acquisition and tracking).
- **Effectiveness of a Particular Spoofing Technique:** Investigate how direct spoofing affects the NEO-7M during the acquisition and the tracking states.
- **Exploring Defense Mechanisms:** Evaluate the resilience of the receiver against spoofing in both states (acquisition and tracking) and propose improvements in anti-spoofing defenses.

[1] The receiver is used on commercial drones. e.g. https://www.flyingtech.co.uk/product/super-mini-ublox-neo-7-gps-for-naze32-cleanflight-betaflight/

- **Document Experimental Findings:** Record and analyze the results of multiple spoofing attempts under specific condition to understand the NEO-7M's defensive capabilities and vulnerabilities.

Chapter 2

Background

2.1 Unmanned Aerial Vehicles (UAVs)

According to Valavanis and Vachtsevanos [49], an unmanned aerial vehicle refers to a pilotless aircraft, a flying machine without an onboard human pilot or passengers, in which "unmanned" implies the absence of a human who pilots the aircraft while onboard. This means that UAV control systems can be onboard or off-board the aircraft. Off-board control refers to the control of the aircraft remotely through a communication link between the aircraft and a control station. On the other hand, onboard control refer to autonomous systems installed on the UAV that allow full control of the aircraft without human intervention. One drawback of an off-board control system is the increased attack surface on the UAV: attackers could potentially target the data link layer between the UAV and the control station and hijack the aircraft. This form of dual attack is discussed in the multi-threat cyber-attacks section in this paper.

2.2 GPS Technology in UAVs

The USA's GPS is one of many Global Navigation Satellite Systems (GNSSs) available for civilian use nowadays. This includes Europe's Galileo, Russia's GLONASS and China's BeiDou[13]. GNSSs, as the name implies, share the same goal of providing positioning, navigation and timing (PNT) services.

GPS is the first GNSS to launch as well as the first to open-source its positioning services globally, which led to its widespread adoption[1]. GPS requires carefully trajected satellites that send signals in specified frequency bands. The GPS constellation consists of at least 24 satellites across 6 equally-spaced orbital planes around Earth to ensure 24-hour availability [34]. According to the Navigation Center of the US Coast Guard [36], there are currently 31 satellites orbiting the GPS constellations. Some of the excess satellites are allocated for redundancy purposes or to ensure reliable service in certain orbital planes to improve accuracy. UAVs utilize the PNT services offered by GPS to ensure smooth navigation and operation. This is a multi-step process that includes signal transmission, signal reception and position computation.

2.2.1 Signal Transmission

UAVs rely on signals from GPS satellites for positioning and navigation. These signals contain information such as the ephemeris of the satellite and the GPS satellite constellation almanac. Whereas the almanac contains a subset of the ephemeris data with reduced precision, the ephemeris contains the precise orbital information of the transmitting satellite [33]. The transmitted GPS signal also contains a time stamp of the satellite's onboard atomic clock, which indicates the exact time the signal was transmitted.This is important as position computation 2.2.3 requires knowledge of precise satellites coordinates.

Signals Frequency Bands When it comes to signal encoding and the frequency bands signals are sent in, some differences arise between each GNSS. Specifically for GPS, there are currently two levels of service: Standard Positioning Service (SPS), for civil use, and Precise Positioning Service (PPS), for military use. As of February 2024, SPS uses 4 signals for civilian use: L1 C/A (coarse acquisition code),

[1]This made it more difficult for other GNSSs to expand in that market. "*The Navigation Satellite System Market is segmented into GPS, GLONASS, Galileo, Beidou, and Others. The GPS segment held the largest market share, accounting for 36% in 2022.*"[37]

L1C, L2C and L5 [32], with the last 3 still being fielded according to the GPS modernization program. The new signals will be of limited use until 18 to 25 satellites get fielded. The names of the signals reflect which frequency band they transmit on; L1 C/A and L1C signals transmit on the L1 frequency band (1,559 - 1,610 MHz), L2C on the L2 frequency band (1,215 - 1,300 MHz) and L5 on the L5 frequency band (1,164 - 1,215 MHz).

As for the PPS, according to NASA, "*The PPS uses the P(Y) code on both the L1 and L2 frequencies. Access to the PPS is restricted to US Armed Forces, US Federal agencies, and selected allied armed forces and governments* [28]."

The signals used in SPS are not encrypted. This is what makes GPS available for civilian use, but it also makes it susceptible to attacks, such as spoofing, due to its open-source nature. Generally, non-sophisticated spoofing attacks fail against PPS signals since the the signal is encrypted, making it inherently resistant to such attacks without relying on other more complicated spoofing methods [5].

2.2.2 Signal Reception

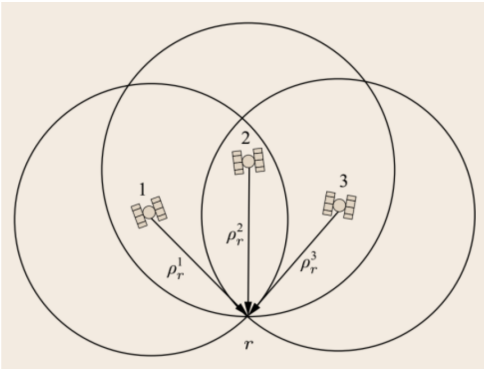
For UAVs to utilize the transmitted GPS signals in their navigation, they need to have GPS receivers installed onboard. It is important to understand how GPS receivers interact with GPS satellite signals under normal conditions in order to realize how they can be misled during GPS spoofing attacks. A typical GPS receiver consists of multiple components including antennas, a front end, a baseband processing unit and an application processing unit [5].

Antenna The main purpose of antennas is to capture the GPS signal. Antennas usually apply amplification and filtration techniques. Antenna amplification enhances the weak GPS signals received. One of the main reasons GPS signals may suffer from power attenuation is the vast distance between the satellite transmitting the signal and the receiver. Filtration by frequency selectivity in the antenna helps the receiver ignore frequencies outside the L band (L1, L2 and L5) to reduce noise and interference from irrelevant sources.

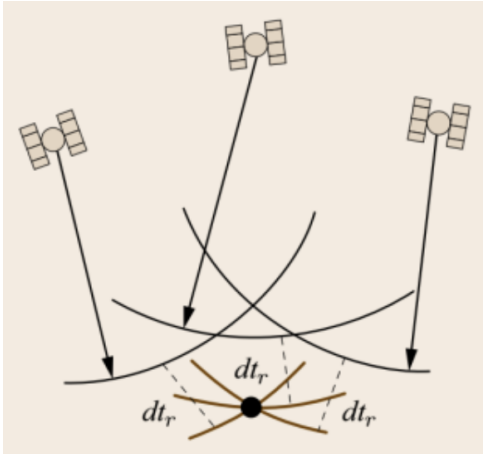
Antennas are also built to boost signals above a certain elevation angle - the angle between the horizon and the satellite with respect to the receiver. Doing so helps the receiver ignore multipath signals [5], which are signals that reach the receiver by reflecting from the ground or buildings. Multipath signals create interference and might introduce phase shifts in the signals which causes the signal to degrade and be less accurate.

The Front End and Baseband Processing The front end and baseband processing unit work together to process the signal delivered by the antenna. The front end filters interfering signals and aim to provide clean signal samples for baseband processing. Unlike the antenna, it applies a more thorough filter by analyzing the signal for out-of-bound interference and noise. After filtration, the front end amplifies and digitizes the signal in preparation for the baseband processing stage. In the baseband processing stage, the digitized signal is decoded to extract data such as the ephemeris, time of transmission, etc. The extracted data is then used to calculate information used to approximate the satellite's coordinates. There are 2 stages that the receiver can be in at the baseband processing step: the acquisition stage or the tracking stage. In the acquisition stage, the receiver searches for GPS satellites to identify which are in view and to extract the Doppler frequency and code delay information from the signal [5]. Extracting these 2 parameters is important for the approximation of the distance between the satellite and the receiver. The code delay represents the time difference between the time of transmission from the satellite and the time of receival in the receiver, and the Doppler frequency represents the change in the signal's frequency due to the motion of the satellite relative to the receiver - as the satellite moves closer to the receiver the signal gets slightly higher and slightly lower otherwise. Once the satellites are identified, the receiver enters the tracking stage. In the tracking stage, the receiver refines the estimates of the signal information to obtain an accurate position, velocity and timing (PVT) solution.

Application Processing The results from the baseband processing stage are then sent to calculate the receiver's position in the application processing stage. The application processing stage calculates the receiver's position by a process called trilateration, which determines the position of an object at unknown coordinates using distances, in this case between the GPS receiver and the satellite(s). The application processing stage is also where this information is applied for a device's intended use-case (i.e. geo-fencing or route planning/optimization).



(a) Receiver position trilateration in a 2D plane.



(b) Clock offset correction.

Figure 2.1: Trilateration and clock offset correction diagrams, taken from [44]

.

2.2.3 Position Calculation

The 4-Satellite Problem As mentioned in the application processing stage 2.2.2, GPS determines an object's position through a process called trilateration (Fig. 2.1a), which requires the distance between the satellite and the receiver. This distance is found by calculating the time difference between the signal's transmission from the satellite and the time of receival in the GPS receiver. Receiver position trilateration requires not just one satellite, but a minimum of 4 satellite signals, to determine the receiver's position. At first, 3 satellites may seem enough to determine the receiver's position, but that does not take into account the timing differences between the satellites and the receiver. Unlike the satellites, receivers do not have atomic clocks, which would synchronise them to the satellites; therefore, an additional 4th satellite signal is crucial for the receiver to calculate both its position and timing error in its own clock. Once the time error is known, the receiver can correctly determine its location as shown in Fig. 2.1b.

The following is an example demonstrating why at least 4 satellites are necessary for 3-dimensional (3D) positioning of a point X:

- A distance between point X and Satellite 1 would place point X on a surface of a sphere.
- Accounting for the distance from point X to Satellite 1 and 2 would restrict the point to lie on a circle formed by the intersection of the spheres of Satellite 1 and 2.
- Accounting for the distance from point X to Satellite 1, 2 and 3 would restrict the point even more so that it could only possibly lie on 2 points.
- Accounting for the distance between point X to Satellite 4 in addition to the other satellites would finally yield 1 unique point where point X would be.

2.3 Software-Defined Radios

More than 30 years ago, radio devices were built with limited functionality for specific purposes determined by their physical components. They were used within a designated frequency band only and use a limited number of modulation schemes, like frequency modulation (FM) and amplitude modulation (AM) methods in old-school radios. Making any changes to these devices back then was technically challenging for non-specialists due to their hardware-centric approach. This made it more difficult for hobbyists and professionals alike to experiment with radio devices for new applications, until software-defined radios (SDRs) were invented. A software-defined radio is a radio device in which certain properties of the device, like the frequency band the device is receiving from or transmitting to, can be tuned using software updates instead of hardware modification, which means that, depending on the hardware used in the SDR's build, an SDR can be tuned to a wide range of frequencies and protocols, like FM radio, cellular networks and WIFI. This offers great flexibility to SDR users and allows more experimentation with radio

devices since they are no longer restricted to a single frequency band or communication standard (e.g. a fixed modulation method). Such adjustable settings of SDRs are the reason they are so appealing - a user can change the radio device's settings without the need to change the hardware.

Despite the capability of SDRs to be adjusted to software updates for their operation, it is worth highlighting that the choice of components significantly affects the performance, flexibility and range of capabilities of the SDR; thus, inspecting the hardware of an SDR is critical to verify its ability to operate for each specific use-case. SDR components can generally be split into digital and analog components. The digital components of SDRs, like field-programmable gate arrays (FPGAs) and signal processors, are used to implement functions of radio devices like signal modulation/demodulation. Analogue components, like amplifiers and antennas, are used to handle the continuous signals that enter or leave an SDR.

To receive and transmit signals, SDRs need analog-to-digital converters (ADCs) and digital-to-analog converters (DACs), respectively . An ADC converts continuous analog signals into digital format in the SDR's receiving path [38]. The ADC does so by sampling and quantizing a signal at discrete intervals, with the sampling rate set by the SDR's hardware design and controlled through software up to the hardware's maximum limit. After sampling and quantizing the signal, the digital representation of the signal can be processed by a computer software in order to demodulate the signal or decode the information it carries. Conversely, a DAC converts digital signals back into analog form for transmission in the SDR's transmitting path[38]. Before transmitting data, SDRs create the digital representation of the signal by modulating and encoding it. The DAC then converts the digital signal to its analogue form by generating continuous waves that mirror the characteristics of the digital signal. The analogue signal is then transmitted through an antenna, usually through an amplifier beforehand to add power to the signal, which is crucial for ensuring the signal is strong enough to travel over the desired distance.

A shared component between the digital and analogue parts is the signal filter [38]. Signal filters allow signals at software-defined frequencies to pass through while attenuating (blocking) signals at other frequencies. When receiving, analogue filters are used before the ADC conversion to select the desired signal and reject out-of-band interference, which are signals that are on frequency bands outside the desired bands, while digital filters further reduce unwanted signals after the ADC conversion. Similarly, transmission in an SDR utilizes the same filters to ensure ideal signal quality. Digital filters are used before the DAC conversion step to modify the bandwidth of the signal and eliminate unwanted noise. Once the signal is digitally filtered, the DAC converts the signal to analogue form, after which the analogue filter ensures the signal is transmitted at the specified power level with the right modulation in addition to further removing unwanted frequencies in the signal.

Some of the key properties of an SDR include its operational mode and sampling rate and depth (or bit depth). An SDR's operational mode describes its communication capabilities - simplex, half-duplex or full-duplex. Simplex communication is just one-way communication. Simplex SDRs offer only 1 mode, either the transmitting or receiving mode. The device can only be used to receive signals or transmit signals, but not both. An example of a simplex radio device is an FM radio receiver, capable of only receiving signals. Half-duplex communication is non-simultaneous bidirectional communication, meaning that half-duplex SDRs can be used to receive or transmit signals, but not both at the same time. The device can switch between modes in its operation, but it can't operate in both modes at the same time. Walkie-talkies are a classic example of half-duplex operation - users can press the button on the walkie-talkie to talk (transmit) or release the button to listen (receive). Full-duplex SDRs are the most capable; they allow bidirectional communication with the ability to receive and transmit signals simultaneously. However, it is necessary to separate the transmit and receive paths in the SDR and implement some isolation techniques to avoid self-interference in the device, which might significantly deteriorate the quality of the signal being received or transmitted. Telephone communication is one clear example of the utilization of the full-duplex operational mode. In the context of GPS spoofing, the operational mode of an SDR is a deciding factor in the type of spoofing attack an attacker might run, with some types of spoofing attacks offering more devious approaches.

Sample rate and sample depth define an SDR's capability to accurately represent radio signals. Sample rate dictates the interval between the samples received or transmitted - it affects the amount and quality of the data. Sample depth affects the dynamic range, the ratio between the highest and the lowest signal amplitudes without distortion, and, thus, the accuracy of the data. Sample depth is not a concept unique to SDRs, but found in various areas like music photography. The more sample depth an SDR has, the more accurately the signal amplitude it receives or transmits is represented. For example, when receiving signals, a 16-bit sample depth SDR would be able to distinguish between subtle differences in signal amplitude whereas an 8-bit sample sdr would miss slightly-varying amplitude data. Higher sample depth improves the detection of weak signals from the noise surrounding it. It might be easier to think of

```
Usage: gps-sdr-sim [options]
Options:
  -e <gps_nav>     RINEX navigation file for GPS ephemerides (required)
  -u <user_motion> User motion file in ECEF x, y, z format (dynamic mode)
  -x <user_motion> User motion file in lat, lon, height format (dynamic mode)
  -g <nmea_gga>    NMEA GGA stream (dynamic mode)
  -c <location>    ECEF X,Y,Z in meters (static mode) e.g. 3967283.15,1022538.18,4872414.48
  -l <location>    Lat,Lon,Hgt (static mode) e.g. 30.286502,120.032669,100
  -L <wnslf,dn,dtslf> User leap future event in GPS week number, day number, next leap second
  -t <date,time>   Scenario start time YYYY/MM/DD,hh:mm:ss
  -T <date,time>   Overwrite TOC and TOE to scenario start time
  -d <duration>    Duration [sec] (dynamic mode max: 300 static mode max: 86400)
  -o <output>      I/Q sampling data file (default: gpssim.bin ; use - for stdout)
  -s <frequency>   Sampling frequency [Hz] (default: 2600000)
  -b <iq_bits>     I/Q data format [1/8/16] (default: 16)
  -i               Disable ionospheric delay for spacecraft scenario
  -p [fixed_gain]  Disable path loss and hold power level constant
  -v               Show details about simulated channels
```

Figure 2.2: GPS-SDR-SIM script options

sample depth as a precision dial: the more the sample depth, the more the decimal places are available for measurement (thus, more accurate values measured).

2.4 GPS-SDR-SIM: GPS Signal Simulator

GPS signals follow a specific format to deliver satellite information to receivers. In a spoofed GPS signal, if the parameters being transmitted in the signal are incorrect, receivers will refuse to lock to the spoofed signal or even identify a spoofing attack is underway against the receiver. GPS-SDR-SIM [9] is a Github repository created by Takuji Ebinuma that generates GPS signals. One of the parameters required for the signal to be generated is an ephemeris file; the file can be found in the following site created by NASA:

https://cddis.nasa.gov/archive/gnss/data/daily/

The site provides GPS broadcast ephemeris (brdc) files on a daily basis. More recent brdc files are preferred when generating the GPS signal in order to avoid incorrect parameters in the signal (too far in the past). A GPS signal's pseudorange and Doppler frequency are two parameters that need to be generated before a spoofer transmits. Pseudorange represents the distance between the A) GNSS satellite at the time of signal emission and B) the receiver antenna at the time of reception. Doppler frequency (or Doppler effect) describes the change of frequency of the signal being transmitted due to the relative motion between a transmitter and a receiver at the time of transmission. Using the ephemeris brdc file, the code generates the simulated pseudorange and Doppler values for satellites currently in view. These values are then used to generate I/Q samples of the GPS signal. I/Q signals allow for efficient transmission of signals, as they consist of two separate signals - In-phase and quadrature signals, hence the name. The in-phase signal is a cosine wave while the quadrature signal is a sine wave; both signals are 90 degrees out of phase of each other. The digitized I/Q samples of the GPS signal can be played using any SDR with transmission capability.

As shown in Fig. 2.2, the script allows users to set multiple settings like the duration of the generated signal. The user is also able to assign a static location or a "user motion" file with a trajectory for the generated signal. Motion files can be either .csv or .txt and display the motion data in multiple formats: National Marine Electronics Association (NMEA), Earth-centered Earth-fixed (ECEF) or Latitude Longitude Height (LLH) formats. Users can also set the sample depth of the signal, which is the number of bits per I/Q sample. The ideal number of bits per sample is dependent on the SDR's available sample depth. When transmitting the generated signal, the user is recommended to set the sampling rate (number of I/Q samples to be transmitted per second) to 2.5MHz or 2.6MHz, depending on the SDR model being used. In the demo/execution chapter (chapter 5), a U-blox receiver will be targeted by a Portapack H2 HackRF One SDR in an attempt to spoof the device. For this model, the repository recommends setting the I/Q sample size to 8 bits per sample when generating the signal and transmitting with a sampling rate of 2.6MHz.

It is worth noting that the GPS-SDR-SIM repository has more than 740 forks [18]. These forks generally implement different functionalities or are tailored for specific SDRs. For example, Multi-sdr-gps-sim [31] by Microtronics is a Github repository based on the work in GPS-SDR-SIM. This repository claims to generate the I/Q samples for the GPS L1 signal "on-the-fly", meaning that the spoofed location could be changed during runtime. Another repository, gps-sdr-sim-realtime by the user gym487 on

Github, supports the generation of the I/Q GPS signal samples with realtime playback in the radio software GNU Radio [18]. These examples show the many options and variations possible for use to run GPS spoofing attacks.

2.5 Related Works

Previous studies exist that display results of running GPS spoofing attacks with different setups. In [17], the authors built a setup with a bladeRF x40 SDR to spoof the signal of 3 devices in both indoor and outdoor environments. The 3 devices were an LG L90 model smartphone, u-blox M8 GNSS Evaluation Kit and a u-blox MAX-7Q. In section 5.3 of the paper, the authors described the setup as consisting of a Raspberry Pi central controller, Python and Javascript scripts, an Arduino sensing unit, and a transmission system using the bladeRF x40. The central controller manages the command and control logic between all components. The scripts are responsible for map downloads, construction of NMEA messages (spoofed coordinates), and initiating/stopping the spoofing signals based on the state of the system. The sensing unit is capable of determining the location and orientation of the target UAV with respect to the built setup. It consists of 2 sensors that determine the UAV location in 3D space such as a Lidar sensor for distance measurements and a 3D accelerometer for UAV angle to the horizontal plane. The unit also contains a magnetometer to measure the setup's orientation and a GPS receiver to determine the setup's location. With all this information, the transmission system, made of a bladeRF SDR and a YAGI directional antenna, can then operate over-the-air and transmit the spoofed GPS signal in the direction of the UAV with a directional antenna. In the paper, the authors display the success of this setup in 6 different scenarios. They targeted 3 receivers in 2 environments - indoor and outdoor. Although some runs of the attack struggled with regards to time taken to spoof, the authors recommended jamming the GPS signal in such cases then running the spoofing attack afterwards. This makes spoofing easier as GPS receivers lose their initial lock on the "truthful" satellites and go into search mode, at which point the spoofed signals would be the strongest and only signals available for the GPS receiver to lock onto.

In [24], the authors attempt to spoof the GPS receiver of a 4.5-kg Hornet Mini UAV helicopter drone from Adaptive Flight Incorporated. The helicopter contains an advanced avionics system and a navigation sensor comprising an inertial measurement unit (IMU), a baro-altimeter and u-blox GPS receiver. The authors did not use a commercially available SDR in their experiment, but rather used a custom-built GPS spoofer to transmit the spoofed signal. It is also worth noting that the demo was carried over a lightweight, cabled (coax cable) connection to the drone's GPS module. They attempted what they defined as an overt GPS spoofing attack. Unlike covert spoofing attacks, in which the spoofer tries to be discreet in order to avoid triggering spoofing detection measures in the target system, overt spoofing attacks entail that the spoofer makes no effort to hide its attempt in attacking the target system. In their experiment, the drone was being controlled by a human operator that instructed the drone to hover at around 8.8 meters above ground level. The spoofer transmitted a signal at

$$\eta = 10\,dB$$

where η represents the ratio of spoofing GPS signal power to authentic GPS signal power.

An η value of 10 dB translates to a signal that is 10 times stronger than the authentic one. The signal initially coincided with the same position of the drone. Upon transmitting these signals, the spoofer gained control of the drone's position and velocity measurements. It is safe to assume that due the huge difference between the spoofed and authentic GPS signal (η = 10 dB), the authentic signals indirectly got jammed. The paper does mention that overt spoofing attacks can cause the Automatic Gain Control (AGC) setting in GPS receivers to force the weak signals below the detection threshold due to the strong spoofed signal. After that, the spoofer showed its control over the drone by inducing a signal that reported the drone to be moving southward. The UAV then responds by rapidly moving northward, assuming that it would go back to its original hovering position and counteract the motion. The UAV, under the human operator's control and after the operator notices the drift from the intended drone path, then enters a GPS-denied mode, where GPS signals will no longer be relied on for navigation. In GPS-denied mode, the UAV utilizes a stateful navigation system to move around. At this point, the spoofer no longer has control over the drone; however, the effects of the false signal from the spoofer still persist - the drone's position is still incorrect even under the other navigation systems. That is because the latest state of the drone's position and velocity before entering GPS-denied mode was influenced by the drone's motion under the spoofer's control. The human operator can attempt to fix the position and

velocity offset caused by the spoofer, but it is difficult to do so due to the limited functionality of the drone under GPS-denied mode. This experiment displays the importance of having spoofing detection algorithms in place to, at least, attempt to prevent such overt attacks.

In [21], the authors gave a detailed walkthrough of their GPS spoofer setup and tested its functionality against a DJI Matrice 100 quadcopter drone. The spoofer apparatus used in their demo consists of a Windows 10 laptop, a BladeRFx40 SDR, a 50 dB attenuator and a GPS antenna. Similar to the software used in the demo for this paper, their experiment used the GPS-SDR-SIM (section 2.4) to generate the I/Q data stream of the modified GPS signal. With this setup, the authors were successful in spoofing the location of the GPS receiver on the drone within 25 meters of the spoofer setup (over-the-air transmission). The same setup was used to successfully spoof the location of a HTC Desire 626 Android smartphone. Both mobile data and Wifi were turned off on the phone during the test since Wifi localization systems can prevent non-sophisticated GPS spoofing attempts against mobile phones [22] [2]. When spoofing the drone, the authors left the drone on the test bench without its propellers. The location of the drone, as reported by the software provided by the manufacturer, was successfully spoofed to both a static location and a large circular path. The drone was reportedly attempting to counteract the spoofed movement of the spoofed circular path as displayed by the power of the propellers during the transmission of the spoofed signal.

Chapter 3

Understanding GPS Spoofing Attacks

Amongst the various threats UAVs face, GPS spoofing stands out in particular. GPS spoofing attacks have been a threat to the GPS system since its inception for military use. These attacks mislead their targets by replicating the same signal structure of GPS satellite signals. However, unlike the authentic signals, spoofed signals deliver falsified information about the satellite positions in space and signal transmission clock timestamp. This can deceive GPS receivers, causing a miscalculation in the receivers' positions and clocks and disrupting navigation systems in UAVs. Such attacks allow attackers to potentially take full control of the aircraft, leading to deviation from the UAV's intended path, unauthorized territory incursion, collision or even crashing.

Fundamentally, GPS spoofing targets GPS receivers' reliance on GPS satellite signals to determine their positions and timings. Spoofed GPS signals maliciously affect receivers by manipulating the code delay and Doppler frequency of authentic signals, thus giving the illusion of truthful positions and times to target receivers [40]. This attack can be carried in two primary methods:

- **Hardware Injection**: This method involves injecting the falsified information via a wired connection to the receiver. Against UAVs, this mode of attack might be less common than wireless attacks, since it requires the attacker to gain physical access to the UAV's receiver, but it could potentially be more devastating due to its direct control over the receiver's signal input.
- **Wireless RF Transmission**: In this mode of attack, the attacker broadcasts the falsified GPS signals via a RF transmitter. It requires no physical access to UAVs and can potentially affect multiple UAVs simultaneously if they are in the attacker's transmission range.

This research focuses on the wireless RF transmission spoofing approach and the technical aspects of the attack, focusing on GPS receiver mechanisms, spoofed signal generation, spoofed signal transmission techniques, favorable conditions for GPS spoofing attacks, possible anti-spoofing defense measures to repel certain kinds of GPS spoofing attacks and strategies that make this form of attack stealthy and difficult to detect.

3.1 GPS Receiver Mechanism

Before generating the spoofed signals, it is important to understand the purpose of ephemeris files, the operational states and the booting modes in GPS receivers. For GPS satellites, the ephemerides files contain clock information and orbital parameters (position, trajectory, etc.) of the satellites that are necessary for receivers to calculate time and location on Earth. A GPS receiver relies on this data from multiple satellites to triangulate its own position and calculate a position, velocity and time (PVT) solution. Due to slight deviations in satellites' orbits and clocks, satellite ephemeris files have to be renewed every now and then; for GPS satellites, that is every 2 hours. This constant update of the ephemeris files ensures that receivers are correctly calibrated and receive accurate satellite information for their PVT calculation.

3.1.1 Acquisition and Tracking States

GPS receivers contain satellite-specific channels, each of which can be in either of two states: acquisition or tracking states [11] [12]. In the acquisition state, each channel searches for its satellite's GPS signal. It searches for the code delay and Doppler frequency of the signal in order to assess whether or not the signal is present. If the signal is detected and the code delay and Doppler frequency information can be estimated from the signal, the channel enters the tracking state; otherwise, the channel remains in the acquisition state. In the tracking state, the channel continuously refines these estimates and monitors the quality of their results. A lock detection algorithm is in place for each of the channels in order for them to correctly follow the incoming signals. Once a signal falls below a certain threshold set by the lock detection algorithm, it is considered lost and the channel responsible for tracking the signal goes back to the acquisition state. At that point, the entire process is restarted.

3.1.2 Booting Modes

As for the booting modes, since the ephemeris files are updated so frequently, GPS receivers start up in different modes depending on factors such as the time since it was last on and the difference in distance between its last registered position and its position at start up time. Generally, GPS receivers have 3 booting options: cold, warm and hot starts.

Mode	Start Time	Data Used	Description
Cold Start	2 to 4 minutes	None (all previous data cleared)	Starts from scratch; acquires ephemeris, seeks satellites. Slow due to lack of any prior positioning data.
Warm Start	∼45 seconds	Last known position, valid almanacs	Utilizes last session's data to estimate satellite positions quickly, less acquisition time than a cold start.
Hot Start	∼22 seconds	Ephemeris, last known position and time	Fastest; uses all available recent data including ephemeris stored in memory to immediately find location.

Table 3.1: Start time measurements provided by Measurement Systems Ltd for an IKE GPS module [30]

Cold starts happen when a receiver starts up after it has been turned off for a long time or moved a great distance, such that all prior information about its last known position, time and satellite data is no longer usable. Since the receiver cannot use previously saved key parameters to start up quickly, it goes into the acquisition state and attempts to download the ephemeris files and collect satellite information from the GPS signals it finds on startup. Cold starts take a long time to complete since they have to go through the entire acquisition-tracking process.

In warm start, a receiver has access to its last position before shutting down and to valid satellite almanacs [12] [11]. The receiver is able to use this information to estimate the satellites in view, their code delays and their Doppler frequency, speeding up the acquisition process.

Hot starts occur when a receiver has been turned off for a short time only. When the receiver starts in this mode, its historical data (ephemeris, last known position, time, etc.) is still valid and stored in memory. This data is used in the acquisition state since it speeds up the time it takes to get a fix for the receiver's location. Unless the receiver's position changes drastically, the receiver can use its stored data for its calculations and gain a fix very fast.

3.2 Spoofed Signal Generation

The first step in spoofing a GPS signal is generating the signal that would carry the falsified information. The aim when generating the signal is to cause the target receiver to lock to the spoofed signal, thus causing an incorrect PVT solution calculation. There are two methods for generating the falsified GPS signal; the spoofer can either create the signal using a signal simulator or use and modify the existing GPS signals:

1. **GPS Signal Simulators**: GPS signal simulators generate the spoofed GPS signal using the satellite data available in the ephemerides files. Once a spoofed location is chosen, the simulator generates the code delay and Doppler frequency of satellites at that location, which are then used to

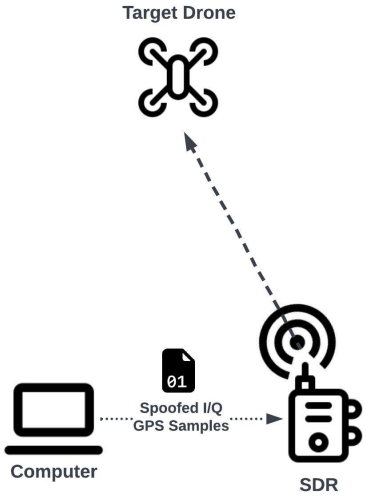


Figure 3.1: Process of generating spoofed GPS signals using a GPS signal simulator and transmitting the signals towards a target drone using an SDR.

generate the falsified signal file. The spoofed signal file then needs to be played via a radio frequency (RF) transmitter. GPS spoofing attacks that utilize signal simulators are called asynchronous attacks because the signals that are generated are not synchronized to the legitimate, up-to-date GPS signals' code delays and Doppler frequency. This is the simplest form of generating the GPS signals, but it is also the most susceptible to detection due its lack of synchronization. During the tracking state in receivers, these signals are viewed as noise, but they can still cause deterioration in the quality of the signals in the receiver [5]. This signal generation method is illustrated in Fig. 3.1 where the generation and transmission process of the spoofed GPS signals are shown.

2. **Repeaters (Receiver-based Spoofers)**: A repeater is a device that receives and transmits the signal it receives. In the context of GPS spoofing, repeaters receive the legitimate GPS signals in order to manipulate them before transmitting them. Since legitimate signals are acquired before retransmission, the signal can simulate the true code delays and Doppler frequency of the satellite at the time of spoofing, hence, they are called synchronous attacks. For instance, the attacker can transmit the spoofed signal after adding external delays to affect its movement. Synchronous attacks make it more difficult for GPS receivers to detect the spoofing attack by simple analysis of the signal properties that would otherwise work against GPS signal simulators. Figure 3.2 provides a detailed view of how a repeater captures and retransmits authentic GPS signals after manipulating them, allowing for a synchronous spoofing attack that mimics legitimate satellite signals.

Signal generation is not the only variable that dictates the success of a spoofing attack. The technique followed when carrying a spoofing attack also plays a big role in the attack's success. Even the simplest signal generation methods, GPS signal simulation for example, can succeed if the proper attack technique is used.

3.3 GPS Spoofing Techniques

GPS spoofing against UAVs presents unique challenges compared to spoofing attacks against other systems, mainly due to their operational nature. While the fundamentals of the spoofing attack - sending falsified GPS signal to a GPS receiver - remains the same for spoofing UAVs and other systems, the nuances in the execution vary against UAVs.

One of the threats of GPS spoofing attacks is their ability to be carried in multiple ways. GPS spoofing attacks can follow certain patterns or accompany the attack with other forms of attack in order to bypass the detection algorithms put in place to stop it. This makes the task of securing GPS receivers and UAVs more difficult, as there are more defences that need to be prepared to avoid takeover from spoofers. The technique a spoofer might decide to implement is affected by multiple factors, such as the environment of the attack, the available information about the target - like its path - and the equipment

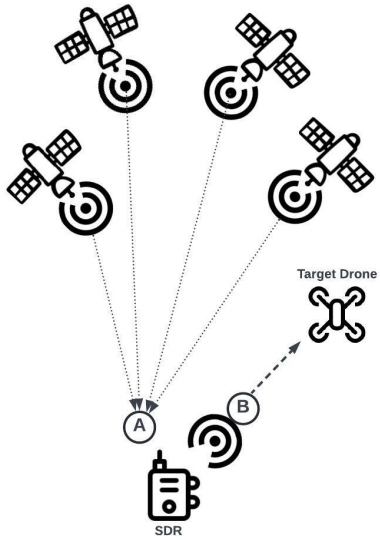


Figure 3.2: Operation of a repeater in a synchronous GPS spoofing attack. Point A represents the reception of the authentic GPS signals. Point B represents the transmission of the spoofed signal.

prepared before the attack. In an over-the-air spoofing attack against UAVs, an attacker can follow the techniques presented in the following list, ordered by their sophistication of operation in ascending order:

- **Direct spoofing**: Direct spoofing attacks are the simplest and the least sophisticated, compared to the other techniques. After preparing the spoofed signal, the spoofer simply transmits the malicious signal with a single source of transmission. In direct spoofing, increasing the transmission power of the spoofing signal can jam the receiver with the spoofing signals themselves. On the opposite end, the power of the signal can be lowered as part of a strategy to avoid detection as discussed in section 3.6 below.

- **Jam and spoof**: As the name suggests, this technique involves two phases - jamming the GPS signal and transmitting the spoofing signal. The main goal is to disrupt the GPS receiver's connection to the authentic signals by overwhelming it with arbitrary, high-power noise at the target frequency. Jamming degrades the quality of the authentic signal reaching the target receiver. The spoofer starts by transmitting the jamming signal in the target receiver's vicinity in an attempt to "drown" the true GPS signals. During the receiver's tracking state, jamming aims to make the receiver lose its lock on the authentic GPS signal due to how weak/noisy they become at arrival, making them fall below the threshold set by the locking algorithm and thus rendering them unreliable. Once the receiver loses its lock, it goes into the acquisition state, at which point it starts "looking" for signals to lock onto.

  The attacker can start transmitting the spoofed signals after jamming, such that when the jamming stops the spoofed GPS signals are the strongest. This can cause the receiver to lock onto the spoofed signals rather than the authentic ones. Such an attack can operate with one or more transmitters. Using one transmitter requires the spoofer to jam the receiver then instantly switch to sending the spoofed signal. If the delay between jamming and spoofing is too big, the receiver might lock on to the authentic signal again. In the case of using two transmitters, the jammer and spoofer can operate simultaneously such that when the jammer is turned off, the receiver can instantly detect the spoofing signal without delay and lock onto it. Another approach when jamming and spoofing simultaneously is balancing the transmission power of the jammer and the spoofer, such that the jammer's signals are strong enough to weaken the authentic GPS signals but weak enough to not heavily deteriorate the quality of the spoofing signals. This approach relies on the fact that the spoofing signals being transmitted are closer to the target receiver than the GPS satellites. Due to the low power of authentic GPS signals, the spoofing signals can withstand the jammer's signals and reach the receiver while the authentic signals would not.

- **Multi-antenna spoofing**: A multi-antenna spoofer setup consists of multiple antennas simultane-

ously transmitting spoofed GPS signals in an area. The spoofer can utilise one or more transmitters as long as there are multiple output sources for the signal. A single transmitter can be connected to multiple antennas, each transmitting its own signal. For example, The BladeRF 2.0 micro series SDRs (xA4, xA5 and xA9) each contain two transmission ports on their board. In SDRs with only one transmission port, like the HackRF One, an extension can be used to increase the number of ports.

For the jam and spoof technique, a Github repository named "JamRF" and authored by Abubakar Ali provides a collection of SDR-ready scripts that enable users to easily run various modes of jamming [4] [3]. The scripts provide a "proactive" jamming technique with multiple modes, tailored to the capabilities of the SDR platform being used to jam. Proactive jamming includes "constant", "sweeping" and "random channel hopping" modes. Constant jamming mode involves continuously emitting a signal at a constant frequency. Sweeping jamming mode is similar to constant jamming but is used to overcome transmission bandwidth limitation in SDRs by "sweeping" across a given frequency range. Random channel hopping mode is an enhanced approach compared to sweeping mode in that it randomly selects frequencies within a given frequency range. These options can help make the jammer less predictable and thus more difficult to counter by jamming defence methods. The open-source nature of these scripts in an accessible format highlights the ease with which such attack techniques can be implemented.

3.4 Attack Conditions: Receiver State, Attack Timing and Environmental Factors

A GPS receiver's susceptibility to spoofing attack is greatly influenced by its operational state - whether it is in the acquisition state or the tracking state. From the spoofer's perspective, UAVs are more vulnerable during the acquisition state as they are searching for satellites to lock to. To target the UAV in its acquisition state, a spoofer can run the attack during the UAV's startup. This is the most optimal time, since the UAV can possibly be undergoing a cold start, especially if it has not been used for a while. In a cold start, the GPS receiver onboard the UAV lacks recent ephemeris information that can help in differentiating spoofed signals from authentic ones. At that point, the receiver does not have any prior knowledge of its location.

Once the UAV's receiver is in the tracking state, the attacker faces more challenges. The receiver might be able to distinguish the spoofed signal from the authentic since it has already loaded recent ephemeris files. The ephemeris data can be used as a reference to verify the integrity of the newly detected spoofed signals. This can potentially lead the receiver to reject the spoofed signals or even identify that it is under attack and notify the user or control station.

The UAV's operational status can also affect the complexity of the attack. When the UAV is stationary, such as when it is booting up, the spoofing attack can be straightforward. The attacker simply starts transmitting the spoofed signal without requiring real-time changes to the signal. When the UAV is stationary, the attacker does not need to account for the movement of the UAV's receiver. However, once the UAV is in-flight, the spoofer has to deal with multiple dynamic factors, such as real-time adjustments of the signal when predicting the UAV's flight path.

Environmental factors can play a role in the effectiveness of the spoofing attack. If the UAV has no clear line-of-sight to the GPS satellites, it may struggle to receive the true GPS signals, which opens an angle of attack for the spoofer. According to [48], even trees can obstruct and weaken GPS signals. In a scenario where a drone is being flown in an urban city surrounded by tall buildings, the "urban canyon" effect can occur. According to [7], "In such situations, if the number of available satellites is less than four at a given epoch, then the complete GPS solution cannot be obtained through the pseudo-range equations [the equations used to calculate the position of the receiver] directly."

On the other hand, a clear line-of-sight between the drone and multiple GPS satellites strengthens the signal, making it more difficult for attackers to spoof the vehicle.

3.5 Defending against GPS Spoofing Attacks

Having explored various GPS spoofing techniques and attacks, it is important to explore the defences designed to protect UAVs against these threats. Defending UAVs against GPS spoofing attacks involves multiple sophisticated mechanisms to detect and mitigate such threats. This section details the defences that can be deployed to protect UAVs, the constraints and challenges with implementing them, and the

methods spoofers might use to avoid detection by some of these defences. Safeguarding UAVs against increasingly sophisticated spoofing attacks is essential to maintain their operational integrity in various environments.

3.5.1 Detection and Mitigation

Signal Power Analysis Considering the weak nature of authentic GPS signals as they travel from satellites in Medium Earth Orbit (MEO) to the surface of the Earth, GPS receiver manufacturers build their receivers with low-noise amplifiers (LNAs). LNAs boost low-power signals without significantly degrading the signal-to-noise ratio - a measure of the power of the desired signal compared to background noise. The NEO-7 GNSS receiver by Ublox, for instance, comprises an LNA to boost the weak signals received from GNSS satellites. Therefore, when an attacker attempts to spoof a GPS receiver at high transmission power, the receiver's detection mechanism can be triggered since the spoofed signal exceeds the expected power range of legitimate GPS signals. This can alert the control station that the receiver is under a jamming or spoofing attack.

Multi-GNSS Receivers Some GNSS receivers support concurrent reception of multiple GNSSs as a feature that enhances the positioning efficiency and capability of the receiver. According to [19], multi-GNSS receivers can offer precise point positioning (PPP) services even in environments with limited satellites in view from a single GNSS system, meaning that satellites operating for different GNSS systems can used together in multi-GNSS receivers to offer PNT services. PPP allows receivers to provide millimetre-level and centimetre-level accuracy in stationary and moving modes, respectively [25]. Apart from the improved positioning feature, multi-GNSS receivers can withstand single-GNSS spoofing attempts, like GPS spoofing. Since GPS spoofing only targets the GPS service, a multi-GNSS receiver can continue to operate under other GNSS services during a GPS attack, even in cases where the high spoofed signal transmission power jams the receiver. Other GNSSs can still operate during jamming since some use different frequencies than the ones used for GPS. As shown in 3.2, in cases where the L1 GPS signal is being jammed, BeiDou and GLONASS satellite systems can still be used for positioning - Galileo E1 cannot be used since it also uses the same frequency as the GPS L1 signal.

Angle of Arrival Defence The angle of arrival (AoA) defence technique is a method that improves signal verification in UAVs. This approach allows receivers using an antenna array to determine the direction from which the GPS signals are arriving [26]. By comparing the phase shift between multiple antennas on a UAV, a GPS receiver can distinguish whether the signal it receives come from ground-level sources or not [23]. This method in particular is effective in detecting spoofing attempts against aerial vehicles since spoofed signals typically originate from ground-level. During warm or hot starts, AoA can also serve as an additional layer to verify the signal's integrity by detecting discrepancies between the expected satellite positions given by the preloaded ephemeris and the positions supplied by the spoofed signal. The AoA defence method can be enhanced through signal processing algorithms and multi-antenna configurations that improve the accuracy of detecting the direction of the incoming signals. As shown in [27], pattern recognition techniques that utilize machine learning can be implemented to reduce false positive occurrences.

Sensor Fusion Not all UAV spoofing detection methods require the defence mechanism to be embedded in the GPS receiver. Spoofing detection mechanisms in UAVs can utilise other components onboard to verify the integrity of the GPS signals. Cross-validation techniques, like sensor fusion (the analysis of data from multiple sensors), can be employed to compare the reported position and velocity from the GPS receiver to other navigation units onboard. An inertial navigation system (INS), for instance, comprises motion and rotation sensors - an accelerometer and a gyroscope, respectively - to navigate a vehicle via dead reckoning, a continuous process that estimates the position of the vehicle given a previously determined position. This allows drones to identify anomalies between the positional data received by a GPS receiver and such units. Detection mechanisms like this rely on the assumption that non-GPS reliant navigation systems are less likely to be compromised, thus serving as more reliable positional data sources.

GNSS	Signal	Frequency in MHz
GPS	L1 C/A	1575.42
	L1C	1575.42
	L2 C	1227.6
	L2 P	1227.6
	L5	1176.45
GLONASS	L1 C/A	1598.0625 – 1609.3125
	L2 C	1242.9375 – 1251.6875
	L2 P	1242.9375 – 1251.6875
	L3 OC	1202.025
Galileo	E1	1575.42
	E5a	1176.45
	E5b	1207.14
	E5 AltBOC	1191.795
	E6	1278.75
BeiDou	B1I	1561.098
	B2I	1207.14
	B3I	1268.52
	B1C	1575.42
	B2a	1176.45
	B2b	1207.14

Table 3.2: Frequency Bands of GNSS Systems [35]

3.5.2 UAV Defence Constraints

UAVs are inherently constrained with their payload weight capacity, which limits the weight of anti-spoofing equipment they can carry. The addition of heavy anti-spoofing equipment not only increases the power consumption of the vehicle but also penalises flight dynamics, such as flight speeds. This can diminish the flight duration of the aircraft and further decrease the maximum payload a UAV can carry. In addition, real-time defence strategies that involve signal processing and anomaly detection require significant processing power, which might put a strain on the limited computing resources on board. Another important factor to consider is cost, especially for commercial UAVs. Defence technologies must be economically viable, otherwise consumer-grade drone manufacturers may find it difficult to integrate these defences in their products. These challenges necessitate functional and economically feasible defence solutions to ensure safe UAV operations.

3.6 Spoofing Strategies to Evade Detection

Spoofers can adopt stealthy and subtle strategies to avoid detection even in non-sophisticated spoofing techniques. To avoid detection, a couple of strategies can be followed by the spoofer.

One approach is for the spoofer to start by transmitting the malicious signal at low power, then increasing it incrementally. This can trick the receiver to lock onto the spoofed signal and circumvent detection mechanisms that rely on signal-to-noise ratio measurements - the spoofed signals are assumed to originate from an actual satellite due to its low power initially. To adjust the transmission power, attackers can use variable RF attenuators, which allow for gradual change in their attenuation level within a given range. The spoofer can start with a high attenuation level and decrease the attenuation in small steps. As demonstrated in [20], this strategy can effectively spoof UAVs and manipulate their flight paths by adjusting GPS and INS data, showcasing the potential success of the attack strategy.

Another spoofing strategy revolves around initially aligning the location transmitted by the spoofed signal with the true location of the drone. The spoofer does so in an attempt to avoid setting any spoofing alarms triggered by abrupt positional changes. Once the drone locks onto the spoofed signal, the spoofer changes the location in small incremental steps, diverting the drone off its intended course without raising immediate suspicion. Over time, the actual location of the drone becomes different to what the drone perceives to be its true location.

Apart from applying these strategies, an attacker can simply use a setup that challenges some of the defences in UAVs. For instance, multi–antenna spoofing setups are direct counters to the AoA defence mechanism in UAVs. By using a network of transmission points, attackers can simulate a number of satellite signals and transmit each from different antennas, thus deceiving the AoA defence [5] [27].

Against multi-GNSS receivers, an attacker can employ the jamming and spoofing technique. The attacker jams multiple GNSS frequencies simultaneously and transmits spoofed satellite signals, each tunes to a specific GNSS frequency (Fig. 3.2), to mislead the multi-GNSS receiver. Such an attack, however, requires precise coordination and higher level of sophistication since GNSS signal structures are not all similar to those of GPS, meaning that each signal needs to be generated separately but transmitted at the same time to spoof the receiver.

Chapter 4

Methodology

This chapter introduces the equipment used in the demo execution in Chapter 5, covering both software and hardware aspects of the setup.

4.1 Portapack H2 HackRF One

For the demo, I used a Portapack H2 HackRF One instead of the plain HackRF One SDR device. The Portapack is just an add-on to the HackRF One that makes it a more portable, standalone device without needing a computer to run the SDR.

4.1.1 HackRF One

The HackRF One SDR is an open-source hardware platform created and manufactured by Great Scott Gadgets. According to the Great Scott Gadgets website [1], the device operates across a wide frequency range starting from 1 MHz up to 6 GHz. This SDR is a half-duplex transceiver, meaning that it can operate to either transmit or receive signals. It handles up to 20 million samples per second, with an I/Q sample depth of 8 bits (8-bit I and 8-bit Q). Its receive (RX) and transmit (TX) gains are software-configurable, allowing users to adjust these settings as per their needs. The SDR features a subminiature version A (SMA) female connector that allows for the attachment of antennas, LNAs, power amplifiers, or Bias-T modules. It includes SMA female inputs and outputs for clock synchronization purposes for scenarios in which multiple RF devices are required to run with precise timing. These ports will not be used in the execution of the demo since only one spoofer device will be used.

4.1.2 Portapack H2

Adding to the features of the HackRF One, the Portapack H2 (Fig. 4.2) transforms the SDR into a fully standalone device. Without the Portapack add-on, the HackRF One requires a power source via USB to microUSB connection with a computer to operate. The Portapack add-on, however, equips the SDR with an embedded rechargeable battery, allowing for mobile use without the need for constant external power. The addition of an LCD touch screen, a user-friendly graphical user interface, simple control buttons and a knob enable users to configure the device without external controls. The Portapack also allows for the

Figure 4.1: A picture of the HackRF One SDR, taken from [1].

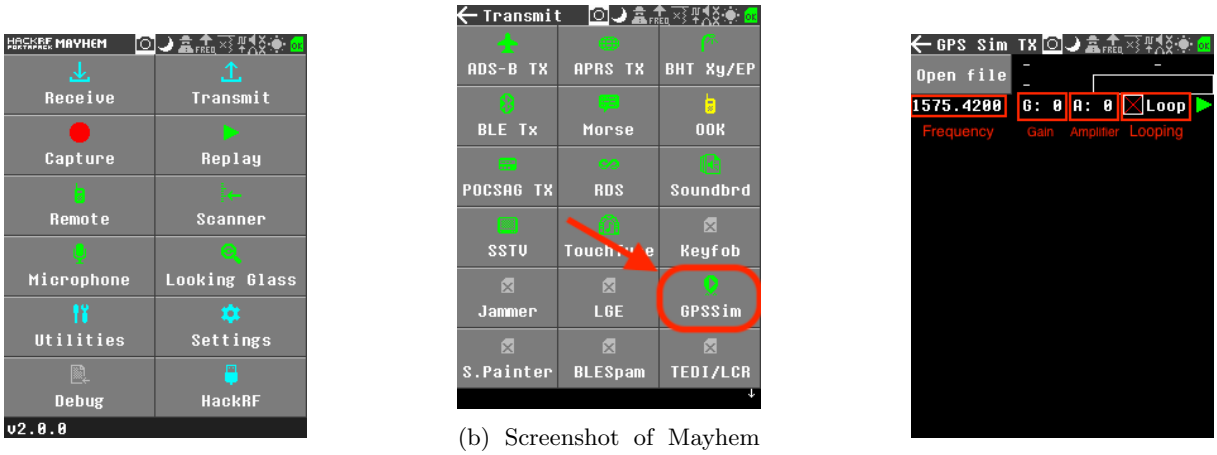
Figure 4.2: Portapack H2 HackRF One

storage of large data files and the execution of various community-developed *'apps'* directly on the device with the addition of a microSD card port. These *apps* simplify can be thought of as similar to libraries used in programming languages - they ease the use of the SDR's capabilities.

4.1.3 Mayhem Firmware and GPSSim App

The Portapack can run on numerous firmware options, such as Havoc, known for its advanced signal manipulation features [15], and Mayhem, known for its many pre-installed utilities and refined GUI [6].

For my purposes, I will be running version 2.0.0 of the Mayhem firmware on the Portapack. After installing the firmware, I downloaded the mayhem_v2.0.0_COPY_TO_SDCARD.zip zip file, which can be found in the release assets of the Mayhem GitHub repository[1], unzipped it and moved the folder to a Samsung EVO Select 128GB microSD card after formatting it to FAT32. The microSD card comes with the exFAT format by default, which cannot be detected by the Portapack.



(a) Screenshot of Mayhem Home page.

(b) Screenshot of Mayhem Transmit page showing GPSSim option.

(c) Screenshot of GPSSim page with labels.

Figure 4.3: Screenshot showcasing Portapack H2 Mayhem firmware display.

The unzipped folder contains the *apps* that extend the functionality of the firmware. More importantly, it contains the GPSSim app that I will be using to transmit the spoofed GPS signal. As shown in Fig. 4.3a, 4.3b and 4.3c, users can start spoofing with a few clicks. To transmit using the GPSSim app, the user can just select a file with the spoofed GPS I/Q samples and click the start button. Transmission settings, such as transmission frequency, gain, power amplifier and loop, can be configured based on the user's needs. Table 4.1 shows the possible options for each setting.

[1] Release v2.0.0: https://github.com/portapack-mayhem/mayhem-firmware/releases/tag/v2.0.0

Setting	Options	Description
Frequency	1 MHz to 6 GHz	Adjustable transmission frequency
Gain (G)	0 to 47 dB	Adjustable gain in 1 dB steps
Amplifier (A)	0 (disabled), 14 (enabled)[2]	Adds 11 dB when enabled
Loop	Disabled, Enabled	Loop through the selected data file

Table 4.1: Setting options in GPSSim page of the Mayhem Firmware.

4.2 Portapack Tx Power Measurement

This section outlines the process of measuring the transmission power of a Portapack using the HP 8594E Spectrum Analyzer (Fig. 4.4. These measurements are crucial for finding the transmission power output provided by the Portapack and ensure accurate operation during demo execution. The values are then validated by referencing the HackRF One documentation and other measurements released online.

4.2.1 Calibration

Before measuring the tx power of the Portapack, the HP 8594E Spectrum Analyzer was calibrated to ensure accuracy. Calibrating the spectrum analyzer is essential to align the measurement tools and get rid of any error in the readings. Given that this device has not been used for more than 3 months, I found this process to be particularly crucial since precision required when measuring small power levels emitted by the Portapack.

To calibrate the spectrum analyzer, I waited a few minutes until the *'OVEN COLD'* label was gone and connected the input and output SMA ports via a coaxial cord. I pressed the *'CAL'* button underneath *'Instrument State'* and then *'CAL FREQ & AMPTD'* to start the automatic device calibration. The calibration was completed after 5 minutes[3].

4.2.2 Measurements

Once the oven was heated and calibrated, I took power measurement while transmitting a sample spoofed signal with the Portapack at the GPS L1 frequency - 1575.42 MHz. The measurements were carried using a coaxial cable directly connecting the spectrum analyzer input port to the Portapack. Two transmission setting configurations were measured to validate the base power level and gain power level increase of the device as per its documentation [16].

Base Power Level Measurement

To measure the base transmission power level of the device, I selected a spoofed signal file on the Portapack's GPSSim page and set the frequency 1575.42 MHz, the gain to 0 and the amplification to 0 (disabled). Table 4.2 describes the settings of the spectrum analyzer during measurement as shown in Fig. 4.5.

[3]Link to the Youtube video I followed to run the calibration: https://www.youtube.com/watch?v=4bDkJl7aI64

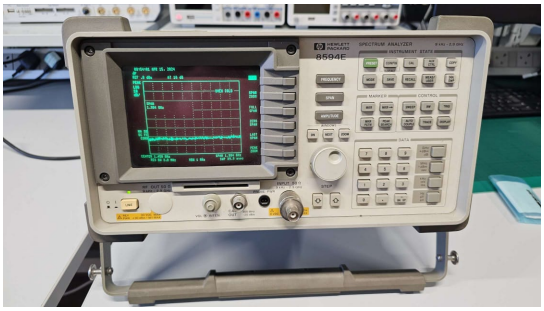


Figure 4.4: HP 8594E Spectrum Analyzer used for power measurements.

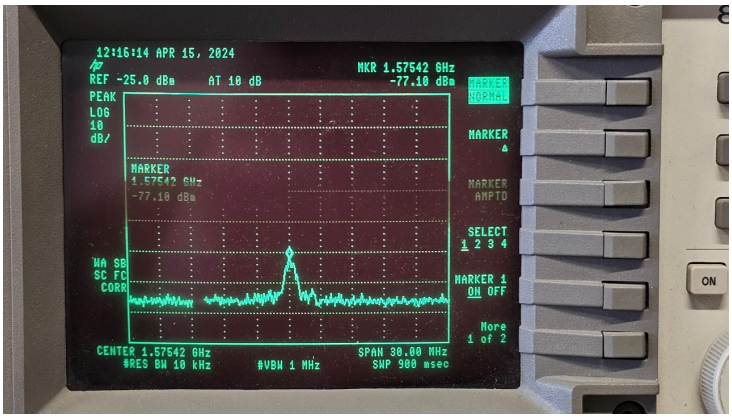


Figure 4.5: The HP 8594E Spectrum Analyzer screen during base power level measurement.

Setting	Value	Description
Reference Level	-25 dBm	The baseline power level against which signal strength is measured.
Center Frequency	1.57542 GHz	Pinpoint the frequency at which Span setting is applied.
Span	30 MHz	The frequency range centered around the Center Frequency.
Video Bandwidth (VBW)	1 MHz	Filter bandwidth. (Affects the displayed noise levels).
Resolution Bandwidth (BW)	10 kHz	The resolution of the frequency components displayed. (Balances sweep time).
Scale	10 dB	Display grid scale per division.
Marker	1.57542 GHz	Placed to measure power at the set frequency.

Table 4.2: HP 8594E Spectrum Analyzer Settings

Results The results of the base power measurement were close to those reported in the HackRF documentation. The peak and the noise floor were recorded at -88.63 dBm and -77.10 dBm. This gives a difference of approximately 11.53 dBm, which is the measured power of the Portapack transmission at the 1.57542 GHz frequency. To verify that the Portapack works as expected, I compared this result to the expected power level reported the HackRF documentation [16]:

- 1 MHz to 10 MHz: 5 dBm to 15 dBm, generally increasing as frequency increases (see this blog post)
- 10 MHz to 2170 MHz: 5 dBm to 15 dBm, generally decreasing as frequency increases
- 2170 MHz to 2740 MHz: 13 dBm to 15 dBm
- 2740 MHz to 4000 MHz: 0 dBm to 5 dBm, decreasing as frequency increases
- 4000 MHz to 6000 MHz: -10 dBm to 0 dBm, generally decreasing as frequency increases

The documentation measured the transmission power to be between 5 dBm and 15 dBm. The Portapack base power result falls within that expected range. Given that my measurement adheres to the documentation's power ranges, the Portapack integrity can be considered verified, and the can be expected to work as intended.

Gain Power Level Measurement

To measure the power increase by the gain and amplification of the Portapack, the transmission gain (G) and amplification (A) settings were set to 30 and 14, respectively. These are the values that were used during demo execution. The same spectrum analyzer configuration shown in Table 4.2 was used. Figure 4.6 shows the spectrum analyzer during the measurement.

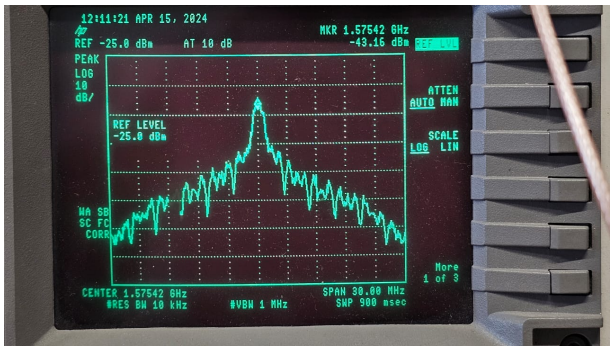


Figure 4.6: The HP 8594E Spectrum Analyzer screen during gain power level measurement.

Results

These peak transmission power and noise floor were subtracted in order to deduce the power increase caused by the gain and amplification settings. The results of the gain power measurement were as follows:

- The noise floor was recorded to be around -89.26 dBm at 1.58855 GHz.
- The peak transmission power was recorded to be around -43.16 dBm at the central frequency of 1.57542 GHz.

With a gain setting of 30 and the amplifier enabled, the total gain was expected to be of 41 dB - the combined values of the two settings (30 dB and 11 dB). Given the recorded values, the difference between the noise floor and transmission power at GPS L1 was approximately 46.1 dB. This gain is close to the expected value and only off by about 5 dB. That difference could have been affected by the differences in the time of recording of the noise floor and transmission peak power. The error is nevertheless minor, suggesting that the theoretical gain values from the HackRF documentation [16] are reliable. These results further verify that the Portapack's components work as expected.

4.3 GPS Antenna Used for Transmitting

Figure 4.7: The Taoglas antenna used in demo execution.

For the demo, the TS.07.0113 RF antenna manufactured by Taoglas Limited was used (Fig. 4.7). This omni-directional antenna, capable of a 180º tilt, operates within a frequency range of 1.561 GHz to 1.61 GHz, making it ideal for GPS applications since it covers the L1 frequency. It comprises an SMA male connector, allowing it to be directly connected to the Portapack's antenna port.

The antenna was used in a straight position without any tilt, positioned in free space. According to the datasheet provided by Taoglas [43], this antenna setup exhibits an average gain of -4.48 dBi (35.65% efficiency), meaning that, compared to an ideal isotropic antenna that transmits equally in all directions, this antenna radiates less power in its main direction of radiation.

4.4 Ublox NEO-7M GPS Receiver

The U-blox NEO-7M GPS receiver is a professional-grade GNSS receiver marketed by U-blox as a cost-effective positioning solution [47]. According to the NEO-7M datasheet [46], the receiver boasts a "*sophisticated RF-architecture and interference suppression* [ensuring] *maximum performance even in GNSS-hostile environments.*"

The NEO-7M supports multiple satellite navigation systems including GPS L1 C/A, GLONASS, and QZSS L1 C/A to enhance positioning and accuracy. The receiver can receive and track either GPS or GLONASS signals, non-simultaneously. The QZSS signals can be received concurrently with GPS to increase signal reliability in the Southeast Asian part of the globe. SBAS, a GNSS augmentation system, satellite signals can also be received to strengthen GPS L1 C/A signals in the receiver. The time to first fix (TTFF) of the NEO-7M as reported in the datasheet vary by booting mode:

- **Cold Start Time**: 30 seconds
- **Warm Start Time**: 28 seconds
- **Hot Start Time**: 1 second

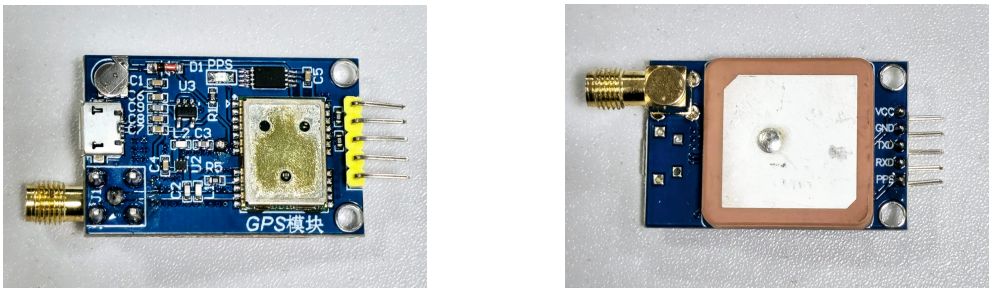
The NEO-7M receiver targeted in the demo execution comprises an embedded passive ceramic antenna, a passive antenna amplifier and a micro-USB connector. To run the receiver for the demo, the receiver was connected to a Macbook Pro 2019 with a micro-USB to USB-C cable. The positioning data was displayed via a script provided by PyGPSClient (Section 4.5).

4.5 PyGPSClient: Displaying Receiver Information

PyGPSClient is an open-source Python project created by *semuconsulting* on GitHub [39]. The project offers a graphical GNSS/GPS testing interface for receivers that support NMEA, UBX, RTCM3, NTRIP and SPARTN protocols. The interface is compatible with both MacOS and U-blox GNSS receivers and can accept GNSS data streams from serial USB connections; therefore, it was used to display the the data produced by the U-blox NEO-7M receiver.

The interface offers view options that offer valuable information about the state of the receiver. The following describes the sections shown in Fig 4.9:

1. **Status Bar**: Displays receiver state information in an easily-to-view format, showing the number of satellites in view (SIV), current timing, latitude, longitude, and altitude positioning data, and updates of the receiver's speed and acceleration.
2. **Console**: Outputs raw data from the receiver. This view allows users to see detailed diagnostics in addition to the navigation data.
3. **Satellite View**: Graphically displays the positions of satellites relative to the receiver, categorized by directions such as north, east, west, and south.
4. **Levels View**: Displays the currently visible satellites and their signal strengths with each satellite's identifier.



(a) NEO-7M GNSS Receiver Module. (b) Ceramic Patch Antenna.

Figure 4.8: Upper and Bottom sides of the U-blox NEO-7M GNSS receiver.

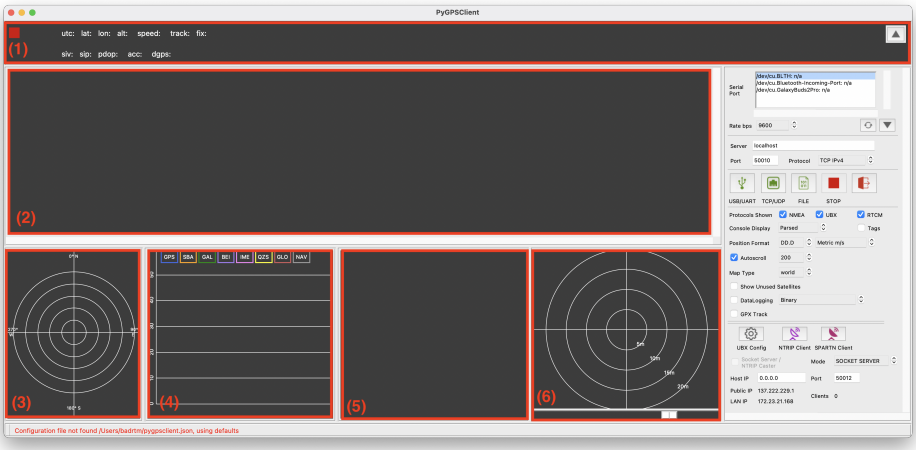


Figure 4.9: PyGPSClient graphical user interface with numbered sections.

5. **Map View**: Displays real-time location of the receiver.
6. **Scatter Plot**: Displays the change in the receiver's position over time relative to its initial fix point.

The NEO-7M was connected to the laptop via a micro-USB to USB-C connection. After being detected by PyGPSClient, the respective serial port can be selected to start processing the receiver data.

4.6 Generating the Spoofing Signals

To test the resilience of the NEO-7M receiver, three different spoofing signals were used: one stationary location (static) and two flight paths (dynamic). These signals aim to evaluate the NEO-7M's response to GPS spoofing attempts to various scenarios.

The spoofing signals were generated using a GPS signal simulator, GPS-SDR-SIM [9] (section 2.4). The static signal file was simply generated using a command line option offered by GPS-SDR-SIM. However, to generate the dynamic signal files, the flight paths had to be created via an external tool. Adhering to the file creation instructions given by the Mayhem firmware GPSSim documentation [29], all signal files had a file extension of (`.C8`) and an accompanying (`.TXT`), each of which used the same filename as their respective (`.C8`) file. All (`.TXT`) files contained the following lines that specify the sample rate and center frequency of the transmission:

```
sample_rate=2600000
center_frequency=1575420000
```

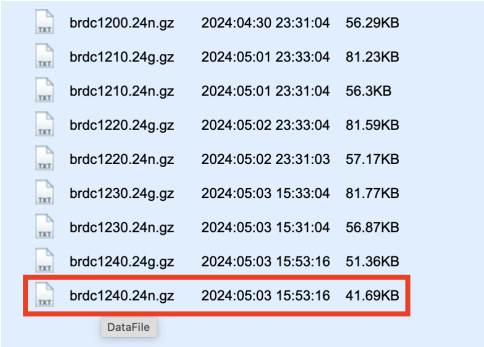
As mentioned in section 2.4, a *brdc* file has to be fed to the script to generate up-to-date simulation files. At the time of execution (3rd of May, 2024), the latest *brdc* file that was uploaded to https://cddis.nasa.gov/archive/gnss/data/daily was 3:53 PM (approximately 1 hour before demo execution). The file, highlighted in Fig. 4.10, was downloaded and used by the GPS-SDR-SIM script to generate the signal files.

4.6.1 Static Signal Simulation

15 July Martyrs Bridge, Istanbul, Turkey

The static spoofing signal was generated using the `-l` command line option provided by GPS-SDR-SIM. The option allows users generate a stationary GPS signal by inputting the desired latitude, longitude, and altitude coordinates directly. The coordinates for the static spoofing signal was chosen to be at `(41.044901, 29.032857, 100)`, which is the 15 July Martyrs Bridge in Istanbul, shown in Fig. 4.11a.

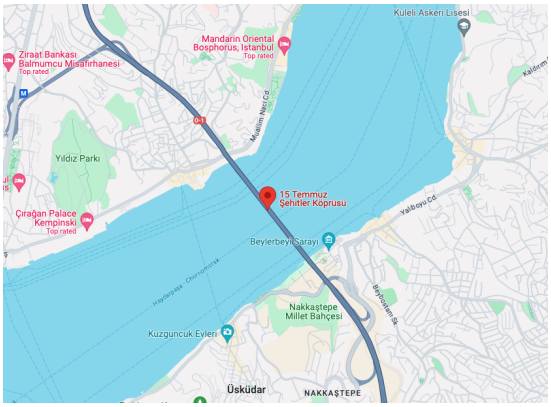
Figure 4.11b shows the script line used to generate the signal and the details of the output file. The duration of the resulting spoofing signal was 300 seconds. The first column after duration refers to the satellites Pseudo-Random Number (PRN) codes, which help identify the satellites being used to simulate the signal. The PRN codes for the static spoofing signal are 5, 7, 9, 13, 15, 20, 22, 27 and 30. The



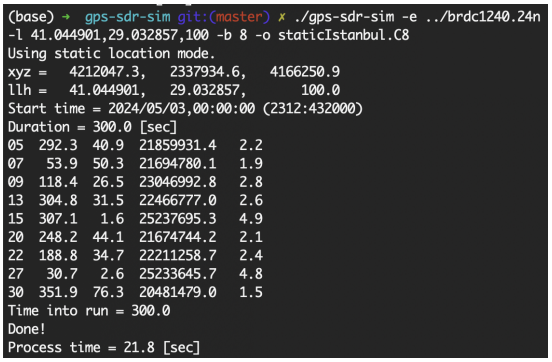
brdc1200.24n.gz	2024:04:30 23:31:04	56.29KB
brdc1210.24g.gz	2024:05:01 23:33:04	81.23KB
brdc1210.24n.gz	2024:05:01 23:31:04	56.3KB
brdc1220.24g.gz	2024:05:02 23:33:04	81.59KB
brdc1220.24n.gz	2024:05:02 23:31:03	57.17KB
brdc1230.24g.gz	2024:05:03 15:33:04	81.77KB
brdc1230.24n.gz	2024:05:03 15:31:04	56.87KB
brdc1240.24g.gz	2024:05:03 15:53:16	51.36KB
brdc1240.24n.gz	2024:05:03 15:53:16	41.69KB

DataFile

Figure 4.10: List of BRDC files. The BRDC file used for demo execution is in the red box.



(a) Spoofed location.



```
(base) → gps-sdr-sim git:(master) ✗ ./gps-sdr-sim -e ../brdc1240.24n
-l 41.044901,29.032857,100 -b 8 -o staticIstanbul.C8
Using static location mode.
xyz =   4212047.3,   2337934.6,   4166250.9
llh =   41.044901,   29.032857,       100.0
Start time = 2024/05/03,00:00:00 (2312:432000)
Duration = 300.0 [sec]
05  292.3  40.9  21859931.4   2.2
07   53.9  50.3  21694780.1   1.9
09  118.4  26.5  23046992.8   2.8
13  304.8  31.5  22466777.0   2.6
15  307.1   1.6  25237695.3   4.9
20  248.2  44.1  21674744.2   2.1
22  188.8  34.7  22211258.7   2.4
27   30.7   2.6  25233645.7   4.8
30  351.9  76.3  20481479.0   1.5
Time into run = 300.0
Done!
Process time = 21.8 [sec]
```

(b) GPS-SDR-SIM command options and output.

Figure 4.11: Spoofing 15 July Martyrs Bridge, Istanbul, Turkey.

details of each satellite are presented from the second to the fifth columns. The columns, starting from the second, represent the satellite's horizontal bearing, elevation angle and pseudo-range and the signal's ionospheric delay, respectively, which are used to assess a satellite's position and the quality of its signals when it reaches the receiver.

4.6.2 Dynamic Signal Simulation

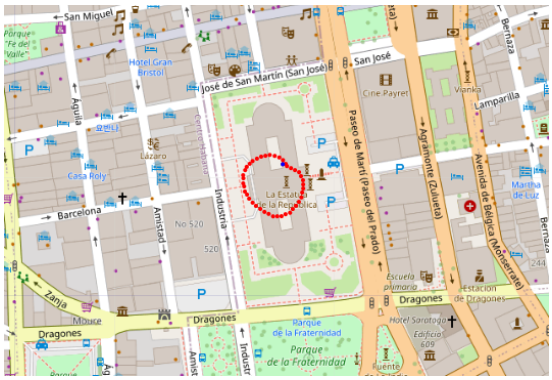
For the dynamic paths, a sequence of Earth-Centered, Earth-Fixed (ECEF) coordinates were used to create the spoofed flight paths, which are a circular path and a zigzag-shaped path. To create the ECEF files, I used ECEF_path_generator [14] created by *fpdf_easytable*, which is an application that generates ECEF coordinates from a drawn path on a map. The output ECEF files were used to generate the GPS signal files using GPS-SDR-SIM.

Circle in Havana, Cuba

This path, generated around the National Capitol of Cuba Museum in Havana, Cuba, is shown in Fig. 4.12a. The circular path was chosen to simulate constant movement around a central point. Figure 4.12b displays the script line used to generate the signal and the details of the output file. The simulated path ended at an altitude of 35 meters above sea level and moved at a speed of 10 km/h, covering a distance of 0.24 kilometers over a duration of 78.1 seconds. The satellites' PRNs are 5, 10, 11, 13, 15, 18, 20, 23, 24, 25 and 29.

Zigzag in Cape Town, South Africa

This path, generated in Cape Town, South Africa, is shown in Fig. 4.13a. The zigzag path was chosen for its constant and abrupt changes in the direction of movement. Figure 4.13b shows the script line used to generate the signal and the details of the output file. The simulated path ended at an altitude of 30 meters above sea level and moved at a speed of 15 km/h, covering a total distance of 0.76 kilometers



(a) Spoofed location and path.

```
(base) ➜  gps-sdr-sim git:(master) ✗ ./gps-sdr-sim -e ../brdc1240.24n
-u ../ECEF_Paths/PATH_Circle.csv -b 8 -o circleHavana.C8
xyz =    780170.8,  -5816184.5,   2490503.4
llh =   23.135149,  -82.360068,        35.0
Start time = 2024/05/03,00:00:00 (2312:432000)
Duration = 78.1 [sec]
05   51.6  25.8  23063147.9   9.8
10  270.2   4.4  25503859.2  19.0
11  121.3   5.9  25099944.8  15.2
13   40.7  35.3  22159240.6   8.3
15   23.7  68.4  20156878.0   5.7
18  334.1  43.1  21789107.9   7.6
20   74.7   7.1  24956134.8  13.6
23  292.9  28.7  23005400.8  10.3
24  146.9  32.5  22467335.4   9.3
25  198.9   1.3  25948669.6  20.0
29  219.6  57.2  20905733.4   6.5
Time into run = 78.1
Done!
Process time = 5.8 [sec]
```

(b) GPS-SDR-SIM command options and output.

Figure 4.12: Spoofing Cape Town, South Africa.

(a) Spoofed location and path, starting from the bottom.

```
(base) ➜  gps-sdr-sim git:(master) ✗ ./gps-sdr-sim -e ../brdc1240.24n
-u ../ECEF_Paths/PATH_ZigZag.csv -b 8 -o zigzapCapeTown.C8
xyz =   5027886.9,   1671083.7,  -3538988.4
llh =  -33.918753,   18.384899,        30.0
Start time = 2024/05/03,00:00:00 (2312:432000)
Duration = 188.4 [sec]
04   99.2   7.3  24920624.4   4.3
06  228.7  67.1  20570813.8   1.6
09   61.7  10.3  24624627.7   4.0
11  280.9  38.2  22110838.0   2.3
12  220.3  14.5  24492610.4   3.7
19  155.2  66.1  20833658.2   1.6
20  332.9  16.3  23991268.9   3.5
22    8.7  49.2  21214041.5   1.9
24  263.4   5.9  25034518.3   4.4
Time into run = 188.4
Done!
Process time = 12.4 [sec]
```

(b) GPS-SDR-SIM command options and output.

Figure 4.13: Spoofing Cape Town, South Africa.

over a duration of 188.3 seconds. The duration of the resulting spoofing signal was 188.4 seconds. The satellites' PRNs are 4, 6, 9, 11, 12, 19, 20, 22 and 24.

Chapter 5

Execution: Spoofing the NEO-7M

This chapter describes the demo execution environment and the results found from running a direct spoofing attack against the U-blox NEO-7M GNSS receiver. The NEO-7M receiver was targeted with the 3 signals generated earlier (1 static signal, 2 dynamic signals) during two different states - acquisition and tracking. Each signal transmission was carried at a separate time from the other (i.e. no overlapping transmissions). During the first state, the receiver did not have a prior fix and was still in its acquisition state. During the second state, the receiver had a fix in Bristol and was in its tracking state.

5.1 Experiment Conditions

The demo was executed with the setup discussed in chapter 4. The tests were carried in a controlled indoor environment to prevent the signal transmission from leaking outdoors. Specific conditions include:

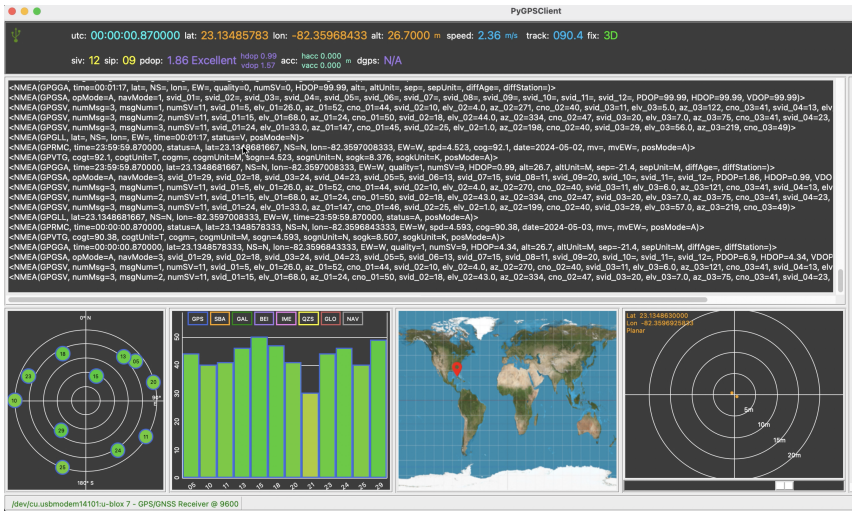
- **Location**: The transmission was conducted in a closed room to contain and minimize RF signal from leaking.
- **Distance and Placement**: The Portapack was positioned approximately 12 meters away from the NEO-7M. Both devices were not moved during transmission of the spoofing signals. Both devices were set on top of two benches at the same elevation.
- **Gain and Amplification**: The Portapack was transmitting with a gain of 30 dB and the amplifier enabled (11 dB) to simulate the direct spoofing technique.
- **Line-of-Sight**: Both the transmitter and the receiver had clear line-of-sight to eliminate any obstructions from interfering with the signal.

5.2 Results

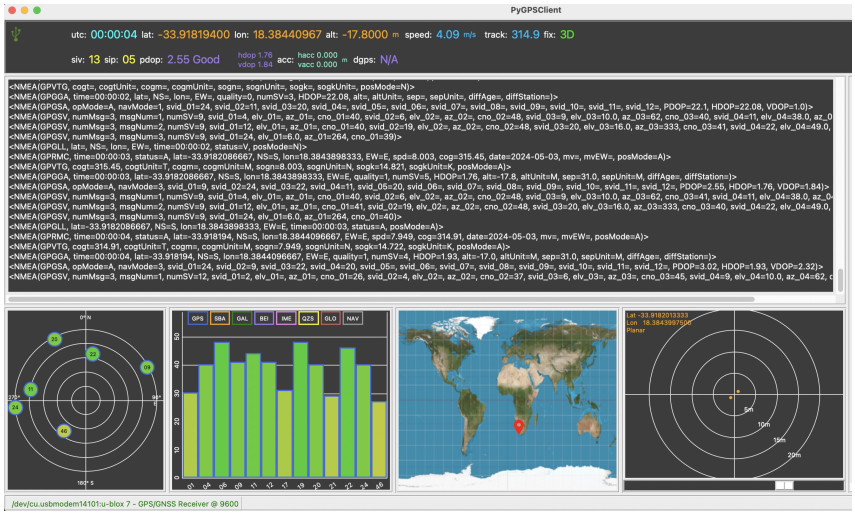
5.2.1 GPS Spoofing during NEO-7M Acquisition State

The three spoofing signals successfully spoofed the NEO-7M receiver during acquisition state. Figures 5.1a, 5.1b and 5.1c show the NEO-7M receiver locking onto the spoofing signals for Istanbul, Havana and Cape Town on the PyGPSClient window. The receiver stayed locked onto all spoofing signals until the end of each of their transmissions. Table 5.1 summarizes the finding.

(a) Initial GPS lock showing the static spoofed location in Istanbul.



(b) Initial GPS lock at the start of the circular spoofing path in Havana.



(c) Initial GPS lock at the start of the zigzag spoofing path in Cape Town.

Figure 5.1: Initial NEO-7M GPS locks to the spoofing signals of Istanbul, Havana, and Cape Town.

Static Spoofing: Istanbul (Stationary)

The static spoofing signal of the 15 July Bridge in Istanbul successfully spoofed the NEO-7M receiver. The spoofing signal was transmitted for only 100 seconds of the full duration (300 seconds). Figure 5.1a shows the PyGPSClient window when the receiver first locked onto the spoofed signals. The receiver was successfully spoofed after 19 seconds of transmitting the spoofing signal. The spoofed location was in Istanbul (`41.044901, 29.032857, 100`), which can be seen in the map view of the window. The *lat* and *lon* variables can be seen displaying the correct spoofed coordinate values; however, the *alt* variable seems to be wrong - at 52.5m instead of 100.0m. The speed value in the status bar was constantly fluctuating between 0 m/s and 0.17 m/s, which led to slight movements reflected in the scatter plot.

Figure 5.2: Scatter plot displaying slight movement of the NEO-7M from the static spoofing signal.

Figure 5.2 shows the scatter plot after 55 seconds since the start of transmission. The scatter plot shows slight deviation in the latitude and longitude positions of the receiver, which was expected to be stationary. The movement can be considered negligible as it was within the 1 meter radius of the specified spoofed position for entirety of the transmission.

Dynamic Spoofing: Havana (Circular Path)

The dynamic spoofing signal around the National Capitol of Cuba Museum in Havana, Cuba, successfully spoofed the NEO-7M receiver. The spoofing signal was transmitted for the full duration (78.1 seconds). Figure 5.1b shows the PyGPSClient window when the receiver first locked onto the spoofed signals. It took only 16 seconds for the receiver to start tracking the spoofed signals. The map view shows the location of the receiver to be in Havana, Cuba. The *lat* and *lon* coordinates of the receiver were around National Capitol of Cuba Museum (`23.135149, -82.360068`) as set by ECEF_path_generator. The altitude of the receiver started at 26.7 meters and slowly made its way to 35 meters in sub-meter steps throughout the signal transmission duration.

The speed measurement shown in the status bar mostly fluctuated between 2.7 m/s and 2.8 m/s, around the speed set during path generation - 10 km/h or 2.78 m/s. Note that the receiver's registered speed hit a low of 2.36 m/s and a high 2.9 m/s at the beginning of the receiver's lock onto the spoofing signals and and the end of the spoofing signal transmission, respectively.

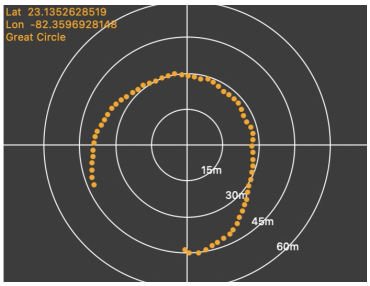


Figure 5.3: The circular path in Havana, displayed in the scatter plot.

Despite not completing the full circle (Fig. 5.3), the scatter plot view clearly demonstrated the circular path generated by the ECEF path generator tool. consistent with the planned spoofing trajectory. The partial circle drawn on the scatter plot seems to be missing around one-fifth of its circumference, which could be attributed to the TTFF of 16 seconds prior to the receiver locking onto the spoofing signal. This time represents around 20% of the total signal duration (78.1 seconds).

Dynamic Spoofing: Cape Town (Zigzag Path)

The dynamic spoofing signal in Cape Town, South Africa, successfully spoofed the NEO-7M receiver. The spoofing signal was transmitted for the full duration, totalling 188.4 seconds. Figure 5.1c shows the PyGPSClient window when the receiver first locked onto and started tracking the spoofed signals, which took only 21 seconds since the start of the transmission of the spoofed signal. As intended by the spoofing signal, the map view in PyGPSClient shows the location of the receiver to be in Cape Town, South Africa, with the *lat* and *lon* coordinate values of the receiver shown on the status bar starting at (`-33.91819400, 18.38440967`). Unlike the dynamic spoofing signal of Havana, the altitude of the receiver was completely incorrect and varied significantly during the course of the transmission. Initially, the altitude was recorded at -15.5 meters when the receiver locked onto the spoofed GPS signal. After 80 seconds of gaining the 3D fix, the receiver's altitude plunged to -28 meters. At the end of the transmission, the altitude had risen to -4 meters.

The speed measurement was stable throughout the spoofed path, correctly hovering around 4.17 m/s as set by the ECEF path generator (15 km/h or 4.17 m/s). However, it was observed that during the sharp turns of the zigzag path, the speed momentarily plunged by 0.15 - 0.56 m/s. Also, it worth noting that the speed of receiver was 4.12m/s when the receiver locked onto the spoofed signal and aggressively fluctuated between 3.61 m/s and 4.39 m/s at the end of transmission.

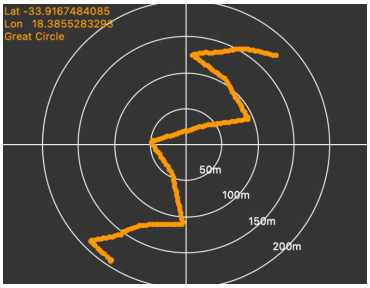
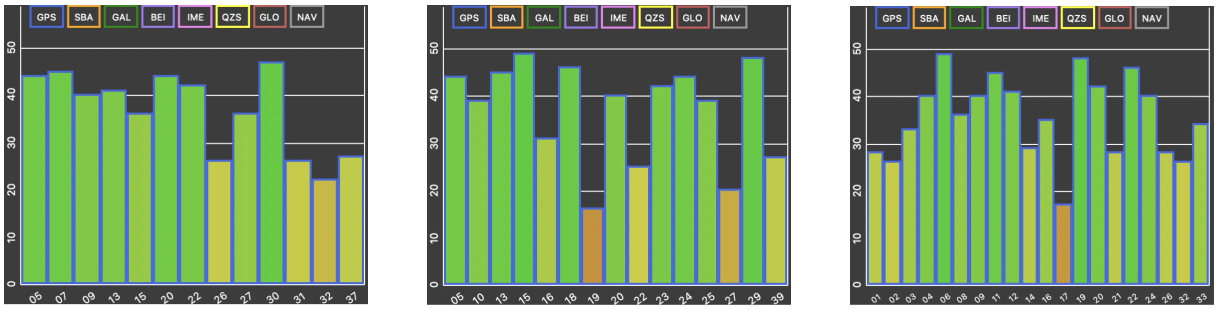


Figure 5.4: Trace of the zigzag path in Cape Town, displayed in the scatter plot.

The zigzag pattern was clearly tracked as show on the scatter plot view (Fig. 5.4). Comparing the path drawn on the scatter plot to the path drawn on the ECEF path generator tool (Fig. 4.13a) shows that the lower half of the zigzag pattern seems to be missing in the scatter plot. Similar to the circular path of the Havana spoofing signal, the missing portion of the path can be attributed to the NEO-7M's TTFF to the spoofing signal since the start of the transmission. The TTFF was only 21 seconds since the start of the transmission, which is about 11.14% of the entire duration of the spoofing signal (188.4 seconds), and that translates clearly to the small missing part of the path in the scatter plot.

Extra Satellites Appearing

An interesting phenomenon that occurred to the NEO-7M receiver during the transmission of each spoofing signals was the appearance of satellite signals that were not generated using the GPS-SDR-SIM script. Figures 4.11b, 4.12b and 4.13b show the terminal screen at the end of generating the simulated GPS signals for Istanbul, Havana and Cape Town, respectively, using GPS-SDR-SIM. The satellites that were identified by the receiver during the transmission of each signal are shown in figures 5.5a, 5.5b and 5.5c. The extra satellites that appeared for both the Istanbul signal and the Havana signal were identified by the receiver before gaining an initial 3D fix. These satellites appeared at the start of the transmission then disappeared after the 3D fix, with the satellites for the Istanbul signal appearing 13 seconds in and the satellites for the Havana signal appearing 10 seconds in. As for the Cape Town signal, the extra satellites were surprisingly detected after the receiver gained a 3D fix, 35 seconds after the start of transmission and around 10 seconds after the 3D fix.



(a) Satellite availability during transmission of stationary spoofing signal in Istanbul.

(b) Satellite availability during transmission of dynamic spoofing signal in Havana.

(c) Satellite availability during transmission of dynamic spoofing signal in Cape Town.

Figure 5.5: Satellite level views of the available satellites detected by NEO-7M during the transmission of the spoofing signals.

	Istanbul (Stationary)	**Havana (Circular)**	**Cape Town (Zigzag)**
Type	Static	Dynamic	Dynamic
Successful Spoofing	Yes	Yes	Yes
Signal Duration	100.0 s	78.1 s	188.4 s
TTFF[1]	19 s	16 s	21 s
Extra Satellites	Yes	Yes	Yes
Extra Satellites Timing	Before 3D fix	Before 3D fix	After 3D fix
Correct Spoofed Altitude	No	Yes	No
Speed	0.00 - 0.17 m/s	2.36 - 2.90 m/s	3.61 - 4.39 m/s
Expected Speed	0.00 m/s	2.78 m/s	4.17 m/s

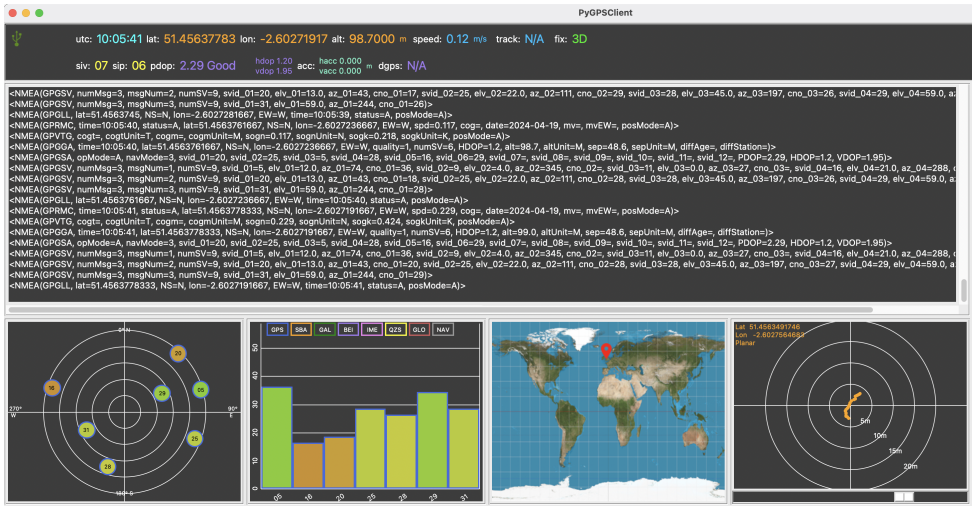
Table 5.1: Summary of GPS Spoofing Results on NEO-7M Receiver in the acquisition state.

5.2.2 GPS Spoofing during NEO-7M Tracking State

After the NEO-7M receiver had successfully been spoofed in its acquisition state by the 3 signals, the receiver was targeted again with the same signals in its tracking state. The receiver was left to lock onto the authentic GPS signals and achieve a 3D fix in Bristol (Fig. 5.6a). The signals transmitted after that point aimed to overwhelm the receiver, causing it to lose its lock on the authentic GPS signals then lock onto the spoofed signals. The same direct spoofing technique used against the receiver in its acquisition state was used against it in its tracking state.

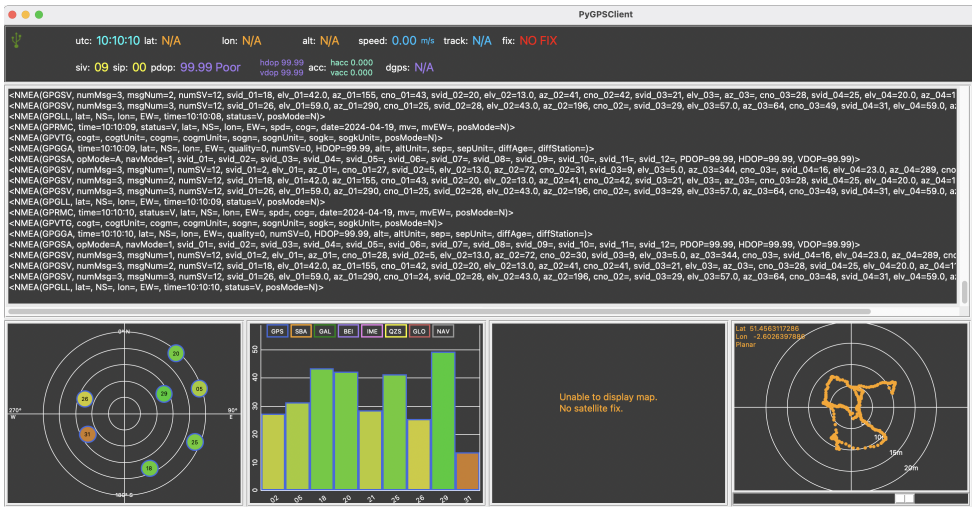
Despite continuous transmission of each spoofing signal, the receiver was never spoofed successfully. However, the receiver was impacted by the spoofing signals in another way. Due to the strength and proximity of the Portapack to the receiver (around 12 meters), the receiver was jammed, leading to a temporary loss of lock to the authentic GPS signals for the duration of the spoofing signal transmission. This result was expected as the first step of spoofing with the direct spoofing technique against GPS receivers in their tracking state is to jam them. After the receiver was jammed, the Portapack continued spoofing till the end of the signal file without success in spoofing the NEO-7M.

These results highlight the effectiveness of the direct spoofing technique against the NEO-7M receiver when it is already locked onto authentic GPS signals. The NEO-7M's resistance to spoofing showcases its successful defense against non-sophisticated spoofing strategies under the aforementioned controlled conditions.



(a) NEO-7M's location fixed in Bristol before spoofing attempt.

(b) Scatter plot indicating the simulated circular path by the spoofing signal, yet the actual location remains unchanged in Bristol.



(c) Scatter plot showing irregular movement during spoofing while the satellite level view still shows strong satellite signals from the Portapack without achieving a lock (jammed receiver).

Figure 5.6: PyGPSClient display showing NEO-7M receiver information during transmission of the Havana dynamic spoofing signal.

Chapter 6

Critical Evaluation

The previous chapter has detailed the execution of and the results observed from running a GPS spoofing attack targeted at the U-blox NEO-7M GNSS receiver. The execution comprised of multiple phases using 3 different spoofing signals during the acquisition and tracking states of the receiver. This experiment aimed to test the NEO-7M's susceptibility to direct GPS spoofing under controlled conditions. This chapter aims to evaluate the effectiveness of the direct spoofing attack technique and the receiver's response to spoofing attacks during the two states. The execution results from these tests highlight the NEO-7M's vulnerability in certain spoofing scenarios and provide valuable insight into the potential defenses that could be implemented to defend against direct spoofing attacks. Furthermore, the results from this experiment are compared with the outcomes of spoofing scenarios in relevant literature in order to measure the NEO-7M's defense performance against other receivers.

6.1 Analysis of the Direct Spoofing Technique

Although the NEO-7M receiver had varying responses to the direct spoofing technique during its acquisition and tracking states, the technique impacted the receiver's performance in both cases. The experiment setup I carried had the receiver placed on a bench similar to [21]. However, in that paper, the receiver was integrated with the drone, and thus readings from other navigational components, like the INU, were used in the drone to analyze the drone's reported position and velocity during the experiment. The spoofing attempts for the stationary and circular spoofing signals were successful similar to the results [21]; however, the authors did not mention the state of the receiver in the drone when running the experiment.

During the acquisition state, all 3 spoofing signals successfully spoofed the receiver and altered its perceived position. In contrast, the same spoofing signals were not successful in spoofing the receiver in its tracking state, but they significantly disrupted its signal reception. Carrying the spoofing attack during these two states of the NEO-7M demonstrates the technique's ability to harm the receiver's integrity and the receiver's defensive capabilities in countering this non-sophisticated spoofing technique.

6.1.1 Spoofing during the Acquisition State

Targeting the NEO-7M in its acquisition state revealed its vulnerability when it lacks a prior 3D fix from authentic GPS signals. Although the direct spoofing technique is the simplest to carry out and the easiest to defend against, the receiver failed to both detect and counter it despite the signals' high transmission power. Although the effect of the strong spoofing signals is not highlighted, directing the high-powered signal towards the receiver was sufficient to easily spoof it without facing difficulties.

Time to First Fix (TTFF)

The TTFF values, detailed in Table 5.1, demonstrate how the NEO-7M was successfully spoofed in under 25 seconds across all spoofing signals. Unexpectedly, the dynamic spoofing signal for Havana was locked onto by the receiver in less time than the stationary spoofing signal for Istanbul, suggesting that the success of a GPS spoofing attack might not depend significantly on type of spoofing signal when the receiver is stationary.

For both dynamic spoofing signals, Havana and Cape Town simulations, the resulting scatter plot on PyGPSClient lacked the beginning of the simulated paths. This was likely due to the time lost before

the receiver locked onto the spoofing signals since the simulated paths were already being sent from the start of the transmissions (Fig. 5.3 and 5.4). This highlights that in non-synchronous spoofing attempts, the delay between the start of the transmission of the spoofing signals and the receiver's TTFF should be accounted for. Therefore, against the NEO-7M, the simulated path can be planned with approximately 25 seconds of arbitrary or little movement at the start in order to produce the desired spoofing results (i.e. to temporally align the simulated path path with the receiver's TTFF).

Speed and Altitude Inconsistencies

The NEO-7M displayed speed inaccuracies during both static and dynamic spoofing attempts. Upon locking locked onto the spoofing signals, the receiver's speed readings fluctuated heavily before stabilising to the expected speed as generated by the ECEF path generator. A similar pattern occurred when the spoofing signal transmissions ended - the reported speed across all signals varied significantly before the receiver lost 3D fix.

For the static spoofing attempt, the NEO-7M provided sub-meter positioning precision (Fig. 5.2) as intended by the stationary simulation in Istanbul. This is an impressive outcome from the The NEO-7M since it does not incorporate PPP, which other models in the NEO-7 series such as the NEO-7P, do [45] [46].

For the dynamic spoofing signals, the speed recorded by PyGPSClient reflected a deviation of no more than 15% below and 5% above the expected speeds generated by the ECEF path generator.

While the altitude measurements for the stationary Istanbul signal and zigzag Cape Town signal were incorrect, the circular Havana signal accurately reflected the expected altitude. This result might have been influenced by the calculations made in the ECEF path generators rather than the signal quality. ECEF calculations need to account for the irregularities of the Earth's surface, which the ECEF path generator might have not accounted for.

Looking at the ECEF conversion function from the ECEF_path_generator codebase [14], the variables a (semi-major axis) and b (semi-minor axis) are defined with constant numbers:

```
toECEF:(function(){
    const a=6378137.0; //mts
    const b=6356752.3; //mts
    var m=(a*a);
    var n=b*b;
    const MPI=Math.PI/180;
    return function(lat, log, h){
        var q=Math.sqrt(m*Math.pow(Math.cos(MPI*lat),2)+n*Math.pow(Math.sin(MPI*lat),2));
        var c=m/q;
        var x=(c+h)*Math.cos(MPI*lat)*Math.cos(MPI*log);
        var y=(c+h)*Math.cos(MPI*lat)*Math.sin(MPI*log);
        var z=((n/m)*c+h)*Math.sin(MPI*lat);
        return roundNumber(x,3)+', '+roundNumber(y,3)+', '+roundNumber(z,3);
    };
})(),
```

Listing 6.1: Conversion from Geographic to ECEF Coordinates

These values assume the Earth is a regular ellipsoid in order to simplify calculations, but they ignore local topographic variations, which might be the reason the altitude reading for Havana was correct while the ones for Cape Town and Istanbul were incorrect. Unlike the ellipsoid model variables, variables for a geoid model could have been used to produce more accurate ECEF coordinates [42] [10]. The shape difference between the two models can be visualized in Fig. 6.1, taken from [8].

Contrary to the geoid model which accounts for Earth's varying topography, the ellipsoid approximates the surface of the Earth, leading to the discrepancies in altitude between the simulated paths' altitudes and the ones observed in by the NEO-7M in PyGPSClient.

6.1.2 Spoofing during the Tracking State

The spoofing attack carried during the NEO-7M's tracking state can more closely describe a direct spoofing attack in a real world scenario, where a GPS receiver would already be in use. Taking the proximity of the Portapack to the receiver in consideration, the direct spoofing technique was expected to jam the receiver in its tracking mode. The receiver was not successfully spoofed by all simulated signals during its tracking state, but the direct spoofing technique did cause the receiver to lose lock to the authentic GPS signals - the receiver was no longer usable at that point.

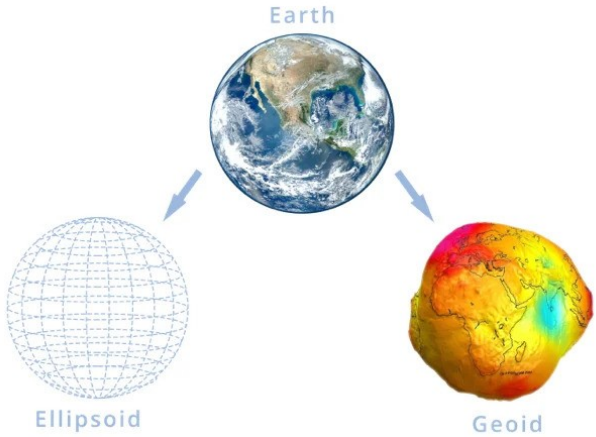


Figure 6.1: Comparison between the ellipsoid and geoid models of Earth. Taken from [8].

Interestingly, the dynamic spoofing signal for Havana interfered with the actual receiver positioning despite the receiver still perceiving its location in Bristol. The map view shows the receiver's perceived location to be in Bristol, but the effects of the simulated circular path in Havana caused the receiver to perceive that it is in moving in a circle, as shown in Fig. 5.6b. The NEO-7M displayed a partial resistance to the spoofing signals as reflected on the scatter plot on the right. The receiver seems to have struggled in tracking the authentic signal, which means that the spoofing signal deteriorated the quality of the tracking in the receiver.

The satellite level view of PyGPSClient shows that less satellites have been detected by the receiver than what was actually sent. When inspecting the satellite PRNs that appeared in the satellite level view, it can be seen that only PRNs 5, 18, 20, 25 and 29 that have been generated by GPS-SDR-SIM for the Havana dynamic signals showed up. PRNs 10, 11, 13, 15, 23 and 24 were never in view even after multiple loops of the signal transmission. Comparing the satellite level views of the dynamic signal for Havana during the acquisition (Fig. 5.5b) and tracking (Fig. 5.6b) states of the receiver quickly shows that some of those satellites are missing when spoofing the receiver in the tracking state. This means that the receiver rejected some of the spoofed signals, possibly due to conflicts between the satellite data it received from the authentic GPS signals and the spoofing signals.

As can be seen prior to the spoofing attempt (Fig. 5.6a), out of the 5 satellite signals that showed up during spoofing, 4 were identified by the receiver (PRNs 5, 20, 25 and 29), meaning that they originated from authentic GPS satellite signals. It is hypothesized that those 4 authentic satellite signals contributed to the receiver's perceived circular motion shown in the scatter plot (Fig. 5.6b). This motion could be interpreted as the receiver attempting to reconcile the authentic signals with the spoofed ones, leading to a movement resembling that of the spoofed signal. Further testing is required to validate this behavior and understand the NEO-7M under spoofing conditions.

Thus, such result proves that apart from jamming the signal, the direct spoofing technique can affect the positioning of the receiver to some degree, even in its tracking state.

Jamming Resistance

The scatter plot displayed in Fig. 5.6c illustrates the NEO-7M's perceived movement prior to losing its 3D fix in Bristol. Despite the fact that the dynamic spoofing signal for Havana was transmitted 3 to 4 times, the receiver maintained its initial lock on the authentic GPS signals. The Portapack's close proximity to the receiver and its high gain setting (approximately 41 dB) created a very challenging environment for the receiver. Nonetheless, it resisted these jamming conditions for a total duration of approximately 4 minutes. Despite the receiver's loss of lock, the satellite level view showed that the receiver was still detecting some satellite signals but did not maintain a lock on any of them, demonstrating the receiver's defensive preparedness against spoofing during its tracking state.

This delay highlights the NEO-7M's ability to differentiate legitimate satellite signals from spoofed ones, given that it has prior knowledge about the GPS satellite constellation and specific details about the satellites it is locked onto (almanac and ephemeris data).

6.2 Defending Against Direct Spoofing Attacks

In addition to the defense mechanisms discussed in Section 3.5, this section provides a few more defenses that can be utilized to hinder the effectiveness of direct spoofing attacks given the observed results from running the attack against the NEO-7M receiver under the aforementioned conditions.

6.2.1 Signal Processing: Detecting Unexpected Satellites

At a certain point during the transmission of the spoofing signals, more satellite signals were received by the NEO-7M than generated by the GPS signal simulator. These additional signals did not seem to have had any impact on the receiver's safety triggers (spoofing detection, for example). As indicated in the satellite level views of PyGPSClient (Fig. 5.5 and 5.6), high-powered signals for multiple satellites were identified when spoofing the receiver in the acquisition state. Most of these signals were momentarily identified by the receiver but then disappeared within a few seconds.

As shown in Table 5.1, during the acquisition state, the receiver detected these unexpected satellite signals before gaining a 3D fix for the static spoofing signal for Istanbul and the dynamic spoofing signal for Havana. Conversely, these signals were detected after the 3D fix in the dynamic signal for Cape Town. In the tracking state, the receiver was resilient and rejected extraneous signals that had not been identified prior to the initiation of the direct spoofing attack.

Regardless of the timing of their appearance, the abrupt presence of high-power signals from satellites serves as a significant indicator of potential spoofing activities. Identifying these strong and non-lasting GPS signals can help the receiver to avoid them - as authentic GPS signals are expected to be weak and consistent - and maintain the integrity of its operation.

6.2.2 Defending Against GPS Spoofing Attacks with Assisted GNSS (A-GNSS)

Assisted GNSS (A-GNSS) is a system that significantly improves receiver's startup time via wireless networks. For the NEO-7M, the U-blox AssistNow Online and AssistNow Offline services can be used not only for that purpose, but to also enhance the receiver's operational resilience [46]. According to the U-blox NEO-7 series documentation, both AssistNow versions are U-blox's own A-GPS services for receivers that can be implemented with not additional hardware or CPU load. The service can function with or without network availability, which is crucial in enhancing the receiver's resilience against GPS spoofing attacks.

Reducing the TTFF is one of the primary defenses against GPS spoofing during the acquisition state. AssistNow Online and Offline allow users to manually upload information, such as ephemeris, almanac, and the last known position, directly to the receiver. This removes the receiver's reliance on external GPS signals for satellite constellation information during startup, thus making over-the-air spoofing attacks ineffective during that state. In addition, speeding up the acquisition process makes receivers to lock onto authentic satellite signals quicker, effectively reducing the window of opportunity for spoofing attempts.

In UAVs, another U-blox service, AssistNow Autonomous, can be used to enable the receiver to function without real-time network connectivity during its flight. The service allows the receiver to locally calculate GPS orbit correction data, removing the need for live updates which might be manipulated by a spoofer.

In the context of the spoofing attacks carried against the NEO-7M, AssistNow services are expected counter the spoofing attempt due to pre-loading the receiver with the satellite constellation data. If the receiver can predict the position of the satellites at startup, then the spoofing attempts would effectively fail, even with the unintended jamming they might cause due to their proximity to the receiver.

Chapter 7

Conclusion

7.1 Future Research Directions

The experiments conducted suggest that anti-spoofing defenses in GPS receivers can still be improved. Further research should be carried in order to understand the behavior of drones under spoofing attacks while they are in-flight.

Despite the partial success of the direct spoofing technique against the NEO-7M, other spoofing techniques should be tested to determine their efficiency, which can be compared by measuring the ease of preparing certain spoofing setups and the effectiveness of the technique being tested in scenarios that mimic real-world attacks more closely. These spoofing scenarios can include the following:

- Targeting the receiver by spoofing its precise position, then gradually shifting position to test the limits of spoofing detection mechanisms.
- Combining spoofing techniques with detection evasion strategies to assess the effectiveness of various spoofing setups.
- Exploring certain spoofing setups against receivers with pre-installed anti-spoofing defenses, such as attempting to spoof the location of a drone that implements the Angle of Arrival (AoA) detection mechanism with a multi-antenna spoofing setup.
- Testing the effectiveness of various low-cost defense mechanisms, such as using multi-GNSS receiver, against non-sophisticated forms of spoofing (since they are the most likely to be faced in the real world due to their ease of operation).

As spoofing techniques get more sophisticated, defensive strategies must also evolve. More research should look into the development of defensive signal processing strategies as they require no extra hardware configurations, possibly making receivers safer and more affordable for integration into commercial drones, where cost-effective options are always preferable.

7.2 Conclusion

While the primary objective of this paper was to evaluate the effectiveness of the direct spoofing technique against drones, certain constraints prevented the execution of tests in the intended manner. Therefore, in my experiment, I tried to replicate the spoofing scenario against a GPS receiver simulating idle drone during startup, a cunning strategy to compromise the drone in its weakest state. This research has highlighted the vulnerabilities and resilience of the U-blox NEO-7M, a popular GNSS receiver for drones, under various spoofing scenarios. The effectiveness of a non-sophisticated GPS spoofing technique - direct spoofing - has been illustrated with multiple types of spoofing signals targeted against the NEO-7M against in both the acquisition and the tracking states. All spoofing attempts against the receiver in its acquisition state were successful, however, all attempts in the tracking state were unsuccessful. Through these experiments, we have gained insights into the behavior of the receiver during a spoofing attack, learned about the operational resilience of the NEO-7M to both high-powered spoofing attempts and identified potential defenses to counter direct spoofing attacks.

GPS spoofing against UAVs presents unique challenges for the spoofer compared to spoofing attacks against other systems, mainly due to their operational nature. While the fundamentals of the spoofing

attack technique - sending falsified GPS signal to a GPS receiver - remains the same for spoofing UAVs and other systems, the nuances in the executions vary. In essence, the success of GPS spoofing attacks against UAVs is a function of good timing, the receiver's state, and environmental conditions, all of which an attacker must consider to successfully carry an attack. This paper demonstrated these concepts from an attacker's point-of-view in order to understand the challenges and further improve anti-spoofing defenses with more clarity in the future.

Bibliography

[1] HackRF one - great scott gadgets. URL: https://greatscottgadgets.com/hackrf/one/.

[2] Security and privacy in communication networks: 17th EAI international conference, SecureComm 2021, virtual event, september 6–9, 2021, proceedings, part i. URL: https://link.springer.com/10.1007/978-3-030-90019-9, doi:10.1007/978-3-030-90019-9.

[3] Abubakar Ali. Jamrf: A collection of jamming scripts for sdr, May 2024. Accessed: 2024-03-16. URL: https://github.com/tiiuae/jamrf.

[4] Abubakar S. Ali, Michael Baddeley, Lina Bariah, Martin Andreoni Lopez, Willian Tessaro Lunardi, Jean-Pierre Giacalone, and Sami Muhaidat. Jamrf: Performance analysis, evaluation, and implementation of rf jamming over wi-fi. *IEEE Access*, 10:133370–133384, 2022. doi:10.1109/ACCESS.2022.3230895.

[5] Leonardo Chiarello. Security evaluation of gnss signal quality monitoring techniques against optimal spoofing attacks, 2018.

[6] Portapack Mayhem Community. Mayhem firmware for portapack. GitHub repository, 2024. Accessed: 2024-02-09. URL: https://github.com/portapack-mayhem/mayhem-firmware.

[7] Youjing Cui and Shuzhi Sam Ge. Autonomous vehicle positioning with gps in urban canyon environments. *IEEE Transactions on Robotics and Automation*, 19(1):15–25, 2003. doi:10.1109/TRA.2002.807557.

[8] DJI. What is datum in surveying?, May 2021. Accessed: 2024-04-26. URL: https://enterprise-insights.dji.com/blog/what-is-datum-in-surveying.

[9] Takuji Ebinuma. GPS-SDR-SIM: An Open Source GPS Simulator. https://github.com/osqzss/gps-sdr-sim, 2017. Accessed: 2024-01-28.

[10] Esri. Understanding the geoid, July 2003. Accessed: 2024-04-26. URL: https://www.esri.com/news/arcuser/0703/geoid1of3.html.

[11] European Space Agency. Lock detectors, May 2024. Accessed: 2024-02-26. URL: https://gssc.esa.int/navipedia/index.php?title=Lock_Detectors.

[12] European Space Agency. Receiver operations, May 2024. Accessed: 2024-05-09. URL: https://gssc.esa.int/navipedia/index.php?title=Receiver_Operations.

[13] European Union Agency for the Space Programme. What is gnss, 2023. URL: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss.

[14] fpdf-easytable Contributors. Ecef path generator. GitHub repository, May 2022. Accessed: 2024-02-09. URL: https://github.com/fpdf-easytable/ECEF_path_generator.

[15] Furrtek. Havoc firmware for portapack. GitHub repository, 2024. Accessed: 2024-02-09. URL: https://github.com/furrtek/portapack-havoc.

[16] Great Scott Gadgets. Hackrf one documentation. GitHub Pages, 2023. Accessed: 2024-01-05. URL: https://hackrf.readthedocs.io/en/latest/faq.html.

[17] João Gaspar, Renato Ferreira, Pedro Sebastião, and Nuno Souto. Capture of UAVs through GPS spoofing using low-cost SDR platforms. 115(4):2729–2754, 2020. doi:10.1007/s11277-020-07211-7.

[18] GPSPATRON. 680 forks on GitHub for GPS signal simulation. Accessed: 2024-03-08. URL: https://gpspatron.com/680-forks-on-github-for-gps-signal-simulation/.

[19] Jing Guo, Xingxing Li, Zhenhong Li, Leyin Hu, Guijun Yang, Chunjiang Zhao, David Fairbairn, David Watson, and Maorong Ge. Multi-GNSS precise point positioning for precision agriculture. 19(5):895–911. doi:10.1007/s11119-018-9563-8.

[20] Yan Guo, Meiping Wu, Kanghua Tang, Junbo Tie, and Xian Li. Covert spoofing algorithm of uav based on gps/ins-integrated navigation. *IEEE Transactions on Vehicular Technology*, 68(7):6557–6564, 2019. doi:10.1109/TVT.2019.2914477.

[21] Eric Horton and Prakash Ranganathan. Development of a GPS spoofing apparatus to attack a DJI matrice 100 quadcopter. 16(1):9. doi:10.1186/s41445-018-0018-3.

[22] Chengbin Hu, Yao Liu, Zhuo Lu, Shangqing Zhao, Xiao Han, and Junjie Xiong. Smartphone location spoofing attack in wireless networks. In Joaquin Garcia-Alfaro, Shujun Li, Radha Poovendran, Hervé Debar, and Moti Yung, editors, *Security and Privacy in Communication Networks*, volume 399, pages 295–313. Springer International Publishing. Series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. URL: https://link.springer.com/10.1007/978-3-030-90022-9_15, doi:10.1007/978-3-030-90022-9_15.

[23] Kai Jansen, Nils Ole Tippenhauer, and Christina Pöpper. Multi-receiver GPS spoofing detection: error models and realization. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 237–250. ACM. URL: https://dl.acm.org/doi/10.1145/2991079.2991092, doi:10.1145/2991079.2991092.

[24] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21513, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21513, doi:10.1002/rob.21513.

[25] Xingxing Li and Xiaohong Zhang. Improving the estimation of uncalibrated fractional phase offsets for ppp ambiguity resolution. *Journal of Navigation*, 65(3):513–529, 2012. doi:10.1017/S0373463312000112.

[26] J. Magiera and R. Katulski. Detection and mitigation of GPS spoofing based on antenna array processing. 13(1):45–57. URL: http://www.jart.icat.unam.mx/index.php/jart/article/view/134, doi:10.1016/S1665-6423(15)30004-3.

[27] Jaroslaw Magiera. A multi-antenna scheme for early detection and mitigation of intermediate gnss spoofing. *Sensors*, 19(10), 2019. URL: https://www.mdpi.com/1424-8220/19/10/2411, doi:10.3390/s19102411.

[28] Catherine G. Manning. Gps, 2023. URL: https://www.nasa.gov/directorates/somd/space-communications-navigation-program/gps/.

[29] Mayhem Firmware Contributors. Gps-sim. GitHub Wiki, May 2024. Accessed: 2024-01-17. URL: https://github.com/portapack-mayhem/mayhem-firmware/wiki/GPS-Sim.

[30] Measurement Systems Ltd. Time to first fix in gps modules, May 2024. Accessed: 2024-04-17. URL: https://www.measurementsystems.co.uk/docs/TTFFstartup.pdf.

[31] Mictronics. multi-sdr-gps-sim. https://github.com/Mictronics/multi-sdr-gps-sim, 2021. Accessed: 2024-01-28.

[32] Navigation National Coordination Office for Space-Based Positioning and Timing. New civil signals. URL: https://www.gps.gov/systems/gps/modernization/civilsignals.

[33] Navigation National Coordination Office for Space-Based Positioning and Timing. Navstar gps space segment, navigation user interfaces, 2012. URL: https://www.gps.gov/technical/icwg/IS-GPS-200G.pdf.

[34] Navigation National Coordination Office for Space-Based Positioning and Timing. How gps works, 2013. URL: https://www.gps.gov/.

[35] NovAtel. Gnss frequencies and signals. https://novatel.com/support/known-solutions/gnss-frequencies-and-signals, 2024. Accessed: 2024-03-17.

[36] Navigation Center of the US Coast Gaurd. Gps constellation, 2024. URL: https://www.navcen.uscg.gov/gps-constellation?order=field_gps_con_plane.

[37] Maximize Market Research. Navigation satellite system market: Global industry analysis and forecast (2023-2029), 2023. URL: https://www.maximizemarketresearch.com/market-report/global-navigation-satellite-system-market/54188/.

[38] Rohde & Schwarz. Understanding radio communication. Rohde & Schwarz News Magazine, 2024. Accessed: 2024-01-03. URL: https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_common_library/dl_news_from_rs/182/n182_radiocomunit.pdf.

[39] semuconsulting. Pygpsclient, May 2024. Accessed: 2024-02-05. URL: https://github.com/semuconsulting/PyGPSClient.

[40] Seong-Hun Seo, Gyu-In Jee, and Byung-Hyun Lee. Spoofing signal generation based on manipulation of code delay and doppler frequency of authentic GPS signal. 19(2):1026–1040. doi:10.1007/s12555-019-0745-6.

[41] Anna Straubinger, Raoul Rothfeld, Michael Shamiyeh, Kai-Daniel Büchter, Jochen Kaiser, and Kay Olaf Plötner. An overview of current research and developments in urban air mobility – setting the scene for uam introduction. *Journal of Air Transport Management*, 87:101852, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0969699719304302, doi:10.1016/j.jairtraman.2020.101852.

[42] Global GPS Systems. Geoids, ellipsoids, and undulation height: Understanding the basics. URL: https://globalgpssystems.com/gnss/geoids-ellipsoids-and-undulation-height-understanding-the-basics/.

[43] Taoglas Limited. *TS.07.0113 Antenna Datasheet*. Taoglas Limited, Global, May 2024. Accessed: 2024-01-27. URL: https://cdn.taoglas.com/datasheets/TS.07.0113.pdf.

[44] Peter J.G. Teunissen and Oliver Montenbruck, editors. *Handbook of Satellite Navigation*. Springer International Publishing, Cham, 2017. Accessed: 2024-02-03. URL: http://link.springer.com/10.1007/978-3-319-42928-1, doi:10.1007/978-3-319-42928-1.

[45] u-blox. NEO-7P DataSheet (UBX-13003787). Online, May 2014. Accessed: 2024-02-24. URL: https://content.u-blox.com/sites/default/files/products/documents/NEO-7P_DataSheet_%28UBX-13003787%29.pdf.

[46] u-blox AG. *NEO-7 Series DataSheet*. u-blox AG, Thalwil, Switzerland, May 2014. Accessed: 2024-02-24. URL: https://content.u-blox.com/sites/default/files/products/documents/NEO-7_DataSheet_%28UBX-13003830%29.pdf.

[47] u-blox AG. *NEO-7 Series Product Summary*. u-blox AG, Thalwil, Switzerland, May 2015. Accessed: 2024-02-24. URL: https://content.u-blox.com/sites/default/files/products/documents/NEO-7_ProductSummary_%28UBX-13003342%29.pdf.

[48] U.S. Government. GPS: The global positioning system - official u.s. government information about the global positioning system (gps) and related topics, May 2024. Accessed: 2024-02-16. URL: https://www.gps.gov/.

[49] Kimon P. Valavanis and George J. Vachtsevanos, editors. *Handbook of Unmanned Aerial Vehicles*. Springer Dordrecht, 2015. doi:10.1007/978-90-481-9707-1.