GPS Software Attacks

Tyler Nighswander
Carnegie Mellon University
Pittsburgh, PA, USA
tylerni7@cmu.edu

Brent Ledvina
Coherent Navigation
San Mateo, CA, USA
ledvina@coherentnavigation.com

Jonathan Diamond
Coherent Navigation
San Mateo, CA, USA
diamond@coherentnavigation.com

Robert Brumley
Coherent Navigation
San Mateo, CA, USA
brumley@coherentnavigation.com

David Brumley
Carnegie Mellon University
Pittsburgh, PA, USA
dbrumley@cmu.edu

ABSTRACT

Since its creation, the Global Positioning System (GPS) has grown from a limited purpose positioning system to a ubiquitous trusted source for positioning, navigation, and timing data. To date, researchers have essentially taken a signal processing approach to GPS security and shown that GPS is vulnerable to jamming and spoofing.

In this work, we systematically map out a larger attack surface by viewing GPS as a computer system. Our surface includes higher level GPS protocol messages than previous work, as well as the GPS OS and downstream dependent systems. We develop a new hardware platform for GPS attacks, and develop novel attacks against GPS infrastructure. Our experiments on consumer and professional-grade receivers show that GPS and GPS-dependent systems are significantly more vulnerable than previously thought. For example, we show that remote attacks via malicious GPS broadcasts are capable of bringing down up to 30% and 20% of the global CORS navigation and NTRIP networks, respectively, using hardware that costs about the same as a laptop. In order to improve security, we propose systems-level defenses and principles that can be deployed to secure critical GPS and dependent systems.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*Security and Protection*

General Terms

Experimentation,Security

Keywords

GPS,Security,RF Attacks

1 Introduction

The Global Position System (GPS) transmits timing information at atomic clock precision to receivers throughout the world. GPS has grown from a limited purpose positioning system to a ubiquitous trusted source for positioning, navigation, and timing (PNT) data. While GPS is commonly known for personal navigation, it is also widely used for precise timing and frequency calibration. For example, cell phone towers (e.g., Verizon) use GPS to calibrate the frequency and timing for transmissions, and the power grid uses GPS to coordinate time stamps for phasor measurements in order to locate power line faults.

All GPS receivers work on the same basic principles to calculate a *navigation solution* in 3D space and time. The navigation solution is calculated by trilateration where the receiver measures its distance from four (or more) satellites: one to resolve each dimension in space-time. Each satellite generates and broadcasts a unique, public pseudo-random number (PRN) stream called the coarse acquisition (C/A) code, which repeats every 1ms. The current time, as determined by an atomic clock on each satellite, week number, and other navigation information is modulated as a *navigation message* on top of the C/A code. GPS receivers generate their own local replica of each satellite's C/A code and estimate the time delta required to align the local replica to the received copy. The time delta, along with the transmission times of each C/A code signal form the distance measurements called pseudoranges. The receiver also decodes the navigation data in order to calculate the satellites' positions and clock offsets. All this information is used to accurately estimate the 3D position and time. In this paper, we focus on the civil GPS signal, which is transmitted without authentication information.

Previous research has focused on two attacks against GPS: jamming and spoofing. Jamming simply transmits noise in the GPS frequency band, preventing a receiver from locking onto the GPS signal. Spoofing attacks are a very specific type of attack that forge the information used to calculate pseudoranges. These attacks are not on the receiver itself per se, as the receiver operates properly just with bogus input data. At a high level, previous work was limited to showing that receivers when given bogus data will output a bogus navigation solution. They did not test for flaws in the actual receiver, or how malicious input flows down to dependent systems.

In this paper, we investigate a larger attack surface against GPS, including the GPS software stack and how errors affect dependent systems. In order to do this, we create new system that allows novel attacks compared to spoofing and jamming. For example, unlike spoofing which misleads and jamming which prevents signal acquisition, we show a 45 second GPS message can disable over 30% of the Continually Operating Reference Station (CORS) network, which is used for safety and life-critical applications. The overall landscape of GPS vulnerabilities is startling, and our experiments demonstrate a significantly larger attack surface than previ-

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

ously thought. In order to help secure GPS systems, we also propose and develop GPS attack defenses not previously considered in the literature.

In particular, we design novel:

1. GPS Data Level Attacks. Previous spoofing attacks are limited to only modifying pseudo-ranges of satellites in view each by some fractional amount. In this paper, we investigate producing good, bad, and wrong data at higher-levels such as the navigation message *in real time with the valid GPS signal.* We call these *GPS data-level attacks* to distinguish them with previous work in spoofing, which was not capable of carrying out attacks of this nature. The advantage is data-level attacks can cause more damage than simple spoofing. For example, we show data-level attacks can remotely crash a high-end professional receiver.

2. GPS Receiver Software Attacks. GPS receivers are computers. Low-end receivers run a basic OS stack (like Windows CE) and simple software. High-end receivers add networking capabilities and complex software, including web-servers and FTP servers, and thus are significantly more complex. We show the software stack can be compromised, in some cases remotely. Since GPS receivers are typically treated as devices, not computers, such vulnerabilities are likely to go unpatched, and represent a serious vulnerability in critical applications.

3. GPS Dependent System Attacks. Higher-level software and systems routinely treat GPS navigation solutions as trusted inputs. We investigate how our attacks at the GPS level can flow up to dependent software. For example, we show that we can permanently de-synchronize the date of Phasor Measurement Units (PMUs) used in the smart grid. We also show we can cause UNIX epoch rollover in a few minutes, and year 100,000 (the first 6-digit year) rollover in about 2 days. These attacks are carried out via RF, showing the attacks can remotely exploit latent bugs that depend upon time and date, that cannot be carried out by spoofers described in previous work which modify only the PRN.

Challenges. Typical software testing assumes generating input for the software is easy. Things are quite different in the RF domain. In particular, in order to test data-level GPS attacks via RF, we need to be able to generate and broadcast our own GPS signal just like a real satellite. Further, receivers have antennas that can distinguish if there are multiple signals, making it potentially possible for a receiver to detect spoofing. Finally, receivers are literally boxes with no programmable API.

Previous work in spoofing has relied upon simulators which cost several thousand dollars and are not suitable for long-range spoofing, or created specialized platforms that performed relatively specialized spoofing attacks against pseudo-range measurements (e.g., [4]). They also have not shown that they can affect the software stack, only that when given garbage input, they will output a correct with respect to the garbage navigation solution.

In order to address the above challenges, we detail the design and development of a novel GPS phase-coherent signal synthesizer (PCSS). The PCSS is like a hybrid receiver and satellite in a box. The PCSS has an input antenna that receives live GPS signals, and outputs malicious signals. In order to be stealthy, our output signal is *phase coherent*, meaning it is in code phase sync with the real GPS satellites. Phase coherence means a receiver won't perceive a difference in phase, making our attacks more stealthy. [1] More importantly, the PCSS hardware and software is designed to allow full programmatic control over the GPS signals in real time. The PCSS allows us to generate good, bad, and malicious GPS data to test the larger attack surface.

[1] An antenna can still distinguish direction. However, launching the attacks from overhead, e.g., via a UAV, is trivial.

The total hardware cost of the PCSS is about the same as a laptop – around $2,500.

Our attacks are applicable in several settings, such as:

1. Manipulate Positioning, Navigation, and Timing (PNT). Planes, cars, trucks, ships, and people all rely on civil GPS to get from point A to point B. GPS is also used extensively for tracking, e.g., the State of California uses GPS-enabled ankle bracelets to track parolees in real time [11]. Cell phone towers, traffic lights, power stations, air traffic control towers, SCADA systems, and other cyber-physical infrastructure use GPS to coordinate activities precisely in the time and frequency domain [20]. Like previous work, our attack can be used to fool receivers into thinking they are somewhere they are not. However, we also show novel attacks that target the data level, such as showing that we can reset the current year of a receiver, e.g., to 2038, the year of UNIX epoch rollover.

2. Manipulate Reference Stations to Amplify Attacks. There are a variety of other critical positioning, navigation, and timing networks that provide PNT information, such as the CORS network [10], Networked Transport of RTCM via Internet Protocol (NTRIP) network [1], and the FAA Wide Area Augmentation System (WAAS). Reference stations maintain a static position p, and also use GPS to calculate their perceived updated position in space-time p'. They can then calculate differential information $p - p'$, which they then use to estimate GPS error to broadcast to other receivers. Reference stations allow dependent receivers to calculate sub-centimeter accurate navigation solutions using civilian GPS. This information is essential to safety and life-critical applications, e.g., WAAS is used in all phases of airplane flight.

We demonstrate spoofing against reference stations, which then amplifies our attack by re-transmitting the faulty information to dependent systems. In addition, *our experiments show we can remotely crash over 30% of the CORS and NTRIP receivers,* which are used in a variety of applications ranging from surveying to unmanned vehicle navigation.

3. Manipulate Down-stream Systems. GPS receivers feed computer systems. Unlike a typical computer, however, there are two access points to attacking a receiver: the RF port and the ethernet port. We show both ports have vulnerabilities which can be exploited. For example, we show data-level attacks via the GPS RF can remotely trigger UNIX epoch rollover, simulate a year 100K rollover in 2 days, and remote ethernet attacks can give us a root shell on the receiver itself.

GPS Defenses. Our findings suggest despite the fact that GPS is an unauthenticated broadcast protocol, current receivers treat any incoming signal as guaranteed correct. Worse, receivers often run full OSes with network services. Together, the possibility of RF and ethernet attacks creates a large attack surface. Previous work has suggested long-term fixes such as adding authentication to the civilian signal, adding new directional antennas, and other changes that require hardware modifications [4, 6, 13, 15]. However, such modifications face potentially lengthy deployment cycles, because significant hardware infrastructure must be changed. While we believe such defenses are necessary in the long term, they (a) don't help protect critical services dependent on GPS in the short term, and (b) don't address attacks at the software level.

We suggest shorter-term defenses along two lines to limit the attack surface. First, in order to defend against data-level attacks, we suggest an Electronic GPS Attack Detection System (EGADS) that can at least warn when an attack is underway, and an Electronic GPS Whitening System (EGWS) that re-broadcasts a whitened signal to otherwise vulnerable receivers. Unlike adding authentication

(e.g., as done with military GPS), EGADS and EGWS would require no changes to already deployed hardware or software. Second, we suggest stronger verification of GPS receiver software to help prevent errors. Third, in order to defend against GPS OS attacks, we recommend regular software updates for IP-enabled devices. GPS receivers have the benefit of running a known set of programs on a fixed hardware platform. As a result, patching should be significantly easier than on a general purpose computing platform.

Contributions Overall, our contributions are:

- A systematic investigation of the attack surface for GPS. We show several new practical attacks that can disable a large percentage of receivers used in critical operations.
- We design and build a PCSS, a novel hardware platform for demonstrating data-level attacks are feasible.
- Novel attacks against GPS software. Although on the one hand all software has bugs, we demonstrate a unique entry point: the RF port.
- We propose, design, and build an electronic GPS attack detection system that does not require changes to existing receivers as a short-term defense. We also make recommendations for systematic changes in the GPS architecture that would increase overall security.

2 The GPS Signal and Previous Attacks

2.1 The GPS Signal

The Global Positioning System is a constellation of nominally 30 satellites in medium-Earth orbit (satellites on occasion fail and are replaced by new satellites, so the exact number at any given time is variable). Each satellite broadcasts ranging signals that can be used by GPS receivers to calculate the receiver's position and the current time. These satellites are controlled by the GPS Control Segment on Earth, which is responsible for estimating each satellite's orbital and clock parameters and uploading this information to the satellites. Each of these satellites has an on-board atomic clock for accurate time-keeping. The satellites broadcast *GPS signals* at two L-band carrier frequencies called L1 (1575.42 MHz) and L2 (1227.6 MHz). Newer satellites also broadcast at the L5 frequency (1176.45 MHz). In this section we recap the basics of the GPS signal; textbooks such as [9] provide more details.

The L1 GPS signal consists of a military signal called Y-code that is modulated in phase-quadrature with a civil signal called the Coarse Acquisition (C/A) code. Some newer satellites are also capable of broadcasting a second military code called M-code. The military Y and M codes are encrypted, which provides a means for military receivers to authenticate the signal. The civil C/A code is not encrypted and does not contain a built-in authentication mechanism. Only special military hardware is capable of receiving the military codes, and details of how to generate those codes are classified. Furthermore, receivers capable of decoding the military signals are relatively expensive, their inner workings a closely guarded secret, and are available to only a very small percentage of users authorized by the US Department of Defense. The vast majority of receivers, including all receivers providing information to US critical infrastructure, network time servers, cell phones, and automobiles, utilize the unauthenticated civil signals. For these reasons, the civil signal is the focus of this work.

Simplistically, signal production begins on the satellite with the generation of a sine wave (also called the carrier) at the broadcast frequency. The L1 carrier is a continuous-wave carrier signal broadcast from each satellite at precisely 1575.42 MHz (ignoring satellite clock frequency error). Bits are then modulated onto this carrier using Binary Phase-Shift Keying (BPSK). In BPSK, bit changes are encoded by switching the phase of the broadcast sine wave by 180 degrees, essentially inverting the broadcast sine wave. Fundamentally, there are two types of information modulated onto this carrier: a high-rate ranging code (1.023 million bps for C/A code) and a low-rate navigation message (50 bps).

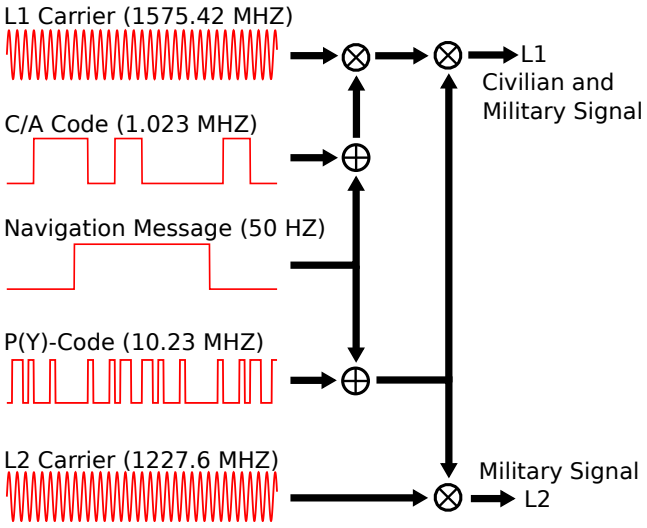

Figure 1: An overview of the GPS radio signal production. ($\oplus$ represents modulo two addition of signals, $\otimes$ represents signal mixing)

The Coarse Acquisition (C/A) ranging code is a time-based pseudo-random number (PRN) sequence unique to each GPS satellite. The PRN codes are quasi-orthogonal, which enables code-division multiple access (CDMA) transmission of the GPS signals. This means that multiple satellites broadcasting at the same frequency but with different PRN sequences can be independently tracked and differentiated from each other. Consistent with CDMA convention, each bit of PRN code is called a chip.

The nominal broadcast time of each chip in the C/A code is well known and can be calculated from the GPS Interface Control Document (ICD) [3]. GPS receivers generate their own local replica of the code broadcast by each satellite and shift it in time until it lines up with the incoming signal. Simplistically, if the receiver has to delay its local PRN generation time relative to the nominal transmit time by t_d for it to line up with the incoming signal from the satellite, then it must have taken the signal that same time t_d to travel from the satellite to receiver. The range from the receiver to the satellite must therefore be $\rho = c \cdot t_d$, where c is the speed of light. Practically speaking, the quantity ρ differs from the true range due to a variety of factors, most significantly the fact that the receiver's clock can not *a priori* be considered well aligned with GPS time (even 1 μs of error in the receiver's clock would equate to 300 m of error in the range measurement). Some other significant errors are satellite clock error and atmospheric delays. Since it differs from the true range to the satellite, the quantity $\rho = c \cdot t_d$ is typically called the *pseudorange*, and it is the fundamental raw measurement made by the receiver.

The purpose of the high-rate PRN ranging code (again, somewhat simplistically) is to allow the receiver to make the pseudorange measurement just described. The purpose of the low-rate navigation data message is to provide information used by higher-level applications in the receiver to derive a navigation solution from the suite of pseudorange measurements (one for each GPS satellite in

view). To do that, the receiver needs to know (1) the position of the satellite at the time of signal transmission, which is encoded in the navigation message and satellite ephemerides, and (2) any deviations in the signal's time of transmit from nominal, which is encoded in the navigation message as satellite clock error coefficients. The message itself is divided into frames and subframes. The specific content of this message is critical to the attacks described later in this paper, and is therefore described in more detail below.

The first navigation subframe contains the time of week in seconds and week number that the navigation data was issued, as well as an identifier for the current set of navigation data called the IODC, information regarding the broadcasting satellite's clock parameters, and the health of the satellite.

The second and third navigation subframes primarily contain the satellite's *ephemeris*. This data provides sufficient information to the receiver to calculate the satellite's position to a few meters or better, and consists of numerous parameters as well as a number used to signify an ephemeris update, called the IODE. While these parameters are not strictly constant, it is worth noting that in normal operation, many of the parameters change only slowly (e.g., update every 2-4 hours). The ephemeris data takes about 30 seconds to receive.

The fourth and fifth navigation subframes primarily contain the *almanac*. This contains location data similar to that found in the ephemeris, but less accurate and for the entire constellation. It also has information regarding predictions of ionospheric conditions which can change the time of flight for a signal traveling from outer space to the Earth. The complete almanac is 25 frames total, requiring about 12.5 minutes total to receive.

In order to form a navigation solution, a receiver will (1) generate pseudorange observables by continuously tracking all in-view GPS L1 C/A code signals and decoding the navigation messages, then (2) solve for receiver position using the pseudorange observables and the information in the navigation message. Typically the receiver has four unknowns – it's X, Y, Z position and its own receiver clock error. Therefore it needs pseudorange measurements from four satellites to solve for these four unknowns.

2.2 Previous Attacks

The concept of GPS spoofing has been known for over a decade in the civil community (and much longer within the military community), with the first public documentation of the spoofing threat occurring in 2001 Volpe report [2]. A key finding of this report predicts most aptly that "[a]s GPS further penetrates into the civil infrastructure, it becomes a tempting target that could be exploited by individuals, groups or countries".

Previous discussed attacks on GPS receivers have included jamming and spoofing. In jamming, significant RF noise is transmitted so that the receiver can no longer pull the satellite signal out of the noise. More interestingly, spoofing attacks generate counterfeit GPS signals that cause the receiver to have incorrect position and time solutions. Since the civil C/A code contains no authentication mechanism and all required signal details are in a public ICD, the generation of these spoofed signals is straightforward.

Note that both of these previous styles of attack target the formation of the receiver's pseudorange, or step (1) from the previous section. They are not attacks on the receiver software itself. Rather, they just give the receiver's software incorrect measurements, yielding an incorrect position solution. Previous work has accomplished these incorrect measurements in two main ways: simple GPS simulator attacks and more sophisticated C/A code spoofing attacks.

Simple attacks use GPS simulators, which are commercial devices used to test the performance of GPS receivers. Simulators generate the L1 signal using either default data, or the user can upload the specification of a signal in a standard format, e.g., Spirent uses NMEA [16]. The main purpose of simulators is offline testing. Simulators for a constellation of satellites typically cost several thousand dollars. In 2002 researchers rented a simulator, added an amplifier, and showed that nearby receivers would lock onto the signal [19].

Simulators are designed to be able to create any realistic suite of GPS signals, and can therefore be programmed to create signals consistent with what a receiver would see at any specific place on the planet. However, they are designed to be connected directly to a receiver for testing purposes. When simply connected to a broadcast antenna, their signals compete with the true satellite signals the receiver was already tracking. Since the simulator has no knowledge nor mechanism to process what is currently being broadcast by the GPS satellites (it only knows what it was programmed to do), its signals will generally be dramatically inconsistent with what the receiver was already tracking. A well-designed receiver can use this inconsistency to detect and reject the spoofed signal.

A more sophisticated C/A code spoofing attack, pioneered in the civil domain by Humphreys et al. [4], contains a GPS receiver that allows it to decode the precise signal being broadcast by satellites in the area. This is then rebroadcast to victim receivers, but with a transmit delay that can be varied relative to the real signal. This causes the satellites to appear to be at a different ranges to the receiver than they really are, causing the receiver to output an incorrect navigation solution. The intriguing thing about this style of attack is that the spoofed signal can originally be made to replicate the broadcast signals perfectly, then gradually moved off target. Since the error is created gradually, this style of attack can be made very difficult to detect. However, it also implies that it takes significant time to create significant navigation errors this way. Note that this style of attack is not fundamentally changing the navigation message, it is just adjusting the apparent pseudorange to the satellite. Our PCSS attack platform is conceptually similar to this style of attack, except that it is designed to allow for live programmatic changes to the GPS data stream.

For stealthy GPS spoofing and satellite lock takeover, several requirements are discussed in depth in Tippenhaurer et al. [17]. Although an attacker can spoof multiple receivers to an arbitrary position (eventually), maintaining the perceived configuration of the receivers is shown to limit the locations from which an attacker can broadcast.

Recall from the previous subsection that the formation of a navigation solution can be divided into two steps: (1) the creation of the pseudorange observables and decoding of the navigation message, and (2) the formation of a navigation solution from the pseudorange observables and the information contained in the data message. Although the GPS spoofing experiments just described are interesting, all work to date has been designed to form incorrect pseudorange observables, targeting Step 1 of the process. Although an incorrect navigation solution is achieved, the software applications inside the receiver are unaffected, and are always doing what they were designed to do – they just received incorrect input and therefore gave incorrect output.

Our approach is unique in that we do not typically target the pseudorange measurements of Step (1) at all (although our system is also capable of that). Instead we focus on targeting the software applications necessary to implement Step (2). The spoofed GPS signal becomes a carrier for a malicious data stream that targets software applications running inside the receiver, exploiting

specific vulnerabilities in those applications. The weaknesses exploited could be in the applications that form the navigation solution, or in downstream applications that interface with or use the navigation solution. They also could be in the applications running on external computers that utilize information obtained over a network from the receiver.

This style of attack is not jamming, nor is it traditional spoofing. This paper documents what we believe to be the first practical cyber attack launched on a system using a spoofed RF GPS signal.

Our attacks are novel over simulators because we can manipulate individual navigation message bits and re-broadcast our attack such that it is a code-phase aligned attack in real time. It is novel over C/A spoofing again because the navigation message can be changed in an arbitrary way. Like existing work [4], we can take over a live satellite lock. There are fewer limitations to our attack over C/A code replay, because we can change the navigation message (ephemeris and almanac) and C/A PRN sequence live.

Another unique aspect of our approach is that it has the potential to be effective across multiple receivers over a large geographic area. To be effective, a single spoofer has to target a specific antenna in a known location [4]. This stems from the fact that it has to produce from one antenna a composite signal that replicates signals from multiple satellites, while still forming a consistent solution at the receiver. Our approach does not suffer from that limitation because the spoofed waveform is really only used as a vehicle to carry the malicious data sequence, not attempting to replicate any specific satellite geometry. Potentially, the only thing limiting the range of our attack's effectiveness, or the number of receivers affected, is the power with which the signal is broadcast and line of sight to the receiving antennas.

3 The Phase-Coherent Signal Synthesizer

The phase-coherent signal synthesizer (PCSS) is a device that simultaneously receives and transmits civil GPS signals. It has been in part derived from the civil GPS spoofer described in [4] and as such, many details about the performance and implementation of the PCSS are similar. It is a software-defined radio (SDR) constructed using commercial hardware as well as hardware and software we developed.

Figure 2: Photograph of the PCSS with its top cover removed

The input analog signal to the PCSS, shown in Figure 2, is the composite of the broadcast GPS satellite signals received at a local antenna. The antenna connects to the RF input to the PCSS. The PCSS also contains an RF output, that one connects to a transmit antenna for local area broadcast or to a coaxial cable that directly connects to the target receiver.

The PCSS acquires and tracks broadcast GPS signals using standard techniques that can be found in [18]. Here, we define *broadcast GPS signals* as the legitimate signals broadcast from the GPS satellites on orbit. The PCSS uses the estimated parameters from each broadcast GPS satellite signal, such as code phase, carrier Doppler shift frequency, and carrier phase, to synthesize new GPS signals from scratch in real time. The synthesized signals also include the modulated navigation data (described in [3]) that is derived either from the received satellite signals, arbitrary user-defined data, or a combination thereof. Additionally, navigation data prediction is used to minimize digital delays within the PCSS that would manifest as a time delay in the navigation data. Minimizing these delays that can be on the order of milliseconds, for example, can improve the stealth of a traditional GPS spoofing attack such as the one described in [4] or our new attacks.

The PCSS hardware consists of (i) radio frequency (RF) down-conversion and up-conversion circuitry, (ii) analog-to-digital (ADC) and digital-to-analog (DAC) converters, (iii) a high-end digital signal processor (DSP), (iv) an FPGA and hardware TCP/IP chip, (v) an embedded microcomputer, (vi) a digitally-controlled attenuator, and (vii) a large amount of flash memory. Figure 3 shows a block diagram of how these different hardware components are connected. The total cost of the commercial-off-the shelf hardware is about $2,500 USD.

These hardware components, along with digital signal processing and estimation software, all written in C++, implement a GPS L1 C/A code receiver and a GPS L1 C/A code signal synthesizer. The receiver tracks 10 GPS C/A code signals and the synthesizer generates 10 GPS C/A code signals in real time.

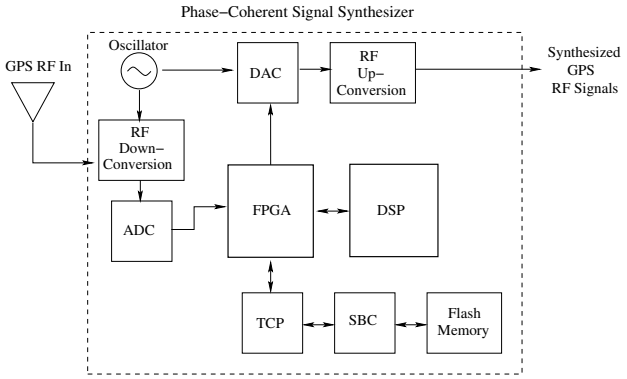

Figure 3: Block Diagram detailing the design and operation of the PCSS

The input RF port on the PCSS receives the broadcast GPS L1 signals from all satellites in view. The RF down-conversion circuitry mixes the composite GPS signal to an intermediate frequency. The ADC then digitizes the signal to 2 bits. Quantizing the composite signal to 2 bits (4 levels) is sufficient for GPS signals, because they are below the noise floor in the L1 band, thus one is simply quantifying the additive Gaussian white noise that, in the worst case, causes loss of ≈ 0.7 dB [18] of the GPS signals. This digital signal is processed by a GPS software-defined receiver running on the high-end Texas Instruments TMS3206455 DSP clocked at 1.2 GHz. The output of the receiver is a set of estimated parameters including the satellites in view, their pseudoranges, carrier Doppler shift frequencies, carrier phases, and the time, position, and velocity of this receiver's antenna. These parameters are input into the GPS signal synthesizer, which is also implemented in software and also runs on the DSP chip. The GPS signal synthesizer takes the input parameters, manipulates them based on the specific attack selected, generates a set of GPS signals containing the time-phased PRN codes and the navigation data, adds them together and then

outputs them to the DAC. The output of the DAC is connected to the RF up-converter that has two stages of up-conversion, with a narrowband band-pass filter between the two stages to eliminate the sampling images. The up-converted signal is centered at the GPS L1 frequency and sent to the output RF port on the PCSS. The digitally-controlled attenuator provides the ability to dynamically control the power level of the output RF signal during an attack.

The PCSS has the same capabilities as the spoofer in [4], but the PCSS has the following improvements:

- The PCSS generates and transmits code-phase coherent GPS signals
- The PCSS intentionally manipulates the information content of the 50bps GPS navigation message
- The PCSS contains internal digital IF data record and playback circuitry and 32 GB of flash memory for storing the data
- The PCSS has an API to control spoofing attacks
- The PCSS transmitter hardware contains a 2 MHz bandpass filter

These improvements are described in more detail below.

First, the PCSS generates code-phase-coherent GPS signals that can be aligned in phase with the satellites' broadcast GPS signals arriving at the PCSS's input antenna. The accuracy of the phase coherency is a small fraction of a GPS L1 C/A code chip, whose duration is nominally 1 microsecond or equivalently 300 meters. Thus, the phase-coherency is typically better than 10 meters. Generating code-phase coherent signals allows the PCSS to target a GPS receiver, possibly at a long stand-off distance, by aligning its generated GPS signals to the broadcast GPS signals that arrive at the target receiver's antenna. This is beneficial, because it is necessary to have the target receiver track and process the PCSS's signals for an attack on the target to be successful. The PCSS output power can be adjusted, e.g., to start at the noise floor and slowly ramp up until a stealthy lock is obtained. A key component to generating code-phase coherent signals is knowing the GPS data bits in the navigation message a priori. Since this is typically impractical, reliable GPS data bit prediction is used. Note that it was recently reported that Wesson et al. have also recently created a spoofer that is code-phase coherent [20].

Second, the PCSS can modify the information content of the GPS navigation message on each generated GPS signal (one for each satellite in view) in real time. To the best of our knowledge, the PCSS is the first to offer this ability. The specific modifications enabled in the current version of the PCSS are the ability to set the square root of the semi-major axis of the GPS satellites' orbits, both in the ephemeris and in the almanac, to an arbitrary number; to replace all of a satellite's ephemeris parameters in subframes 1-3 with that of another satellite; to set the GPS week number in subframe 1 to an arbitrary number; to set the IODE and IODC parameters for each satellite to arbitrary numbers; and to set the leap seconds in subframe 4, page 18 to an arbitrary number. Modifying any parameter in the navigation message typically requires real-time calculation of a new 6-bit parity code in the (32,26) Hamming code for each word. Furthermore, since the navigation message implements a bit-inversion process that is a function of the previous word's last bit (except for word 3), this has to be taken into consideration too.

Third, the PCSS contains an API that allows for programmatic modification in real time of the phase of the GPS PRN codes and the navigation message data. Specific attacks are created using the API, and can be selected to run via a web interface.

Fourth, the PCSS has the ability to record to an internal flash drive either the input digital IF data stream containing the broadcast GPS signals or the output digital IF data stream containing the synthesized GPS signals. Recorded signals can be played back to a target receiver by the PCSS. Playback corresponds to streaming of the digital data at the correct cadence to the DAC and then up-conversion to the GPS L1 frequency. The PCSS contains a small FPGA and hardware TCP/IP IC that connects to both the ADC and DAC. The digital signal to be recorded can be selected and then transferred via TCP packets to the embedded computer and then stored on a 32 GB flash drive. Having this record capability embedded in the PCSS is convenient. This is because the generation of new GPS signals is based on top of the live broadcast GPS signal, which practically means no two sets of synthesized signals are the same. Having the ability to record and then play back the signals provides the ability to identically repeat an experiment.

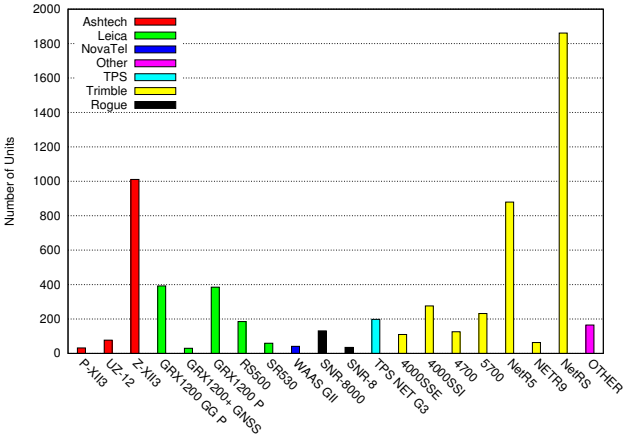

Figure 4: Distribution of receivers on CORS network

Fifth, the PCSS uses a two-stage RF up-conversion process with a 2-MHz bandpass filter to remove artifacts of the digital sampling process. This filter helps remove information in the synthesized GPS signal that could be used to distinguish a spoofer's signals from a legitimate GPS satellite signal.

4 Methodology

Receivers Investigated We investigated attacks against seven different receivers. Each of these receivers have their own purposes and therefore security considerations. The receivers span from consumer grade, costing a few hundred dollars, to professional grade, costing up to about $17,500.

Our consumer receivers investigated are the *Magellan eXplorist 310*, *Garmin eTrex Legend HCX*, *GlobalSat ND-100S*, *uBlox EVK-6H*, *LOCOSYS 23060* and *iFly 700*. The first two devices are handheld receivers primarily targeted for personal navigation, for example hiking or biking, while the iFly is used by private pilots for navigation. The LOCOSYS chip is an OEM device consisting of a GPS antenna and chipset. It was borrowed from researchers who were using it inside quadcopter UAVs. The GlobalSat is a packaged version of the SiRF-III GPS chip, used widely for high-end consumer devices. The uBlox is an evaluation kit for the uBlox GPS chip, which is sold for industrial and automotive applications. Both the eXplorist and iFly receivers run on Windows CE, while the eTrex receiver runs on a custom operating system.

The first professional receiver is the *Trimble NetRS*. The NetRS is intended to be used as a reference station, with a retail cost of about $17,500. The NetRS is widely used in safety-critical settings, such as the CORS network (operated by the United States National Geodetic Survey) and the NTRIP network. We picked the NetRS because it is by far the most popular deployed reference station in

the CORS (about 30% of all receivers, see Figure 4) and NTRIP (1126 out of 3963 stations surveyed, or about 22% of all receivers) networks. The NetRS runs Linux on a PPC chip, provides an ethernet port, and runs several network services such as a web server (for configuration and displaying data), an FTP server, and a NTP server.

The second professional receiver is an *Arbiter 1094B Substation Clock*, which is marketed for use as an accurate time source for Phasor Measurement Units (PMUs) in power stations. PMUs are intended to monitor the power phase at multiple points in the power grid, with any change in phase between two PMUs at the same time indicating a change in power flow. The Arbiter clock aims to be synchronized to within 100ns of GPS time so that PMU readings across geographically distant regions are synchronized. We bought this unit for $1,350, which is the price of similar timing receivers. This specific receiver was chosen due to its high accuracy, serial communications capabilities, and because it is advertised for use within the power grid.

Note that all receivers calculate the navigation solution in firmware, which is closed-source. The main implication is that receivers act as a black box: we often have no idea how it is processing data, or why a fault may occur. Open-source daemons such as gpsd read the navigation solution via a serial link protocol from the appropriate device driver. If the firmware was open-sourced, we could potentially use classic software testing strategies (e.g., symbolic execution, model checking, and fuzzing [2]). Thus in our study the PCSS was absolutely essential since we are effectively only given black box access to the GPS via the RF port.

Testing All our GPS RF attacks are transmitted via the PCSS over coaxial cable using an SMA port. For the professional receivers, no modification was required, as each has ports for external antennas. However, each of our consumer receivers had to be retrofitted with an SMA antenna port through which our signal could be broadcast. This modification required disassembling the GPS receiver, removing the internal antenna, and soldering on an SMA port. Although the iFly receiver had an external antenna port available, we also replaced it with an SMA port for easier interfacing.

The GPS signal reaching the surface of the earth is approximately -160 dBW (1×10^{-16} watts), roughly the equivalent of viewing a 25-watt light bulb in Japan from Los Angeles [19]. Broadcasting at any power would likely affect receivers outside the authors' control, and besides being dangerous, would also potentially be a violation of US law. Although using the coaxial cable has the unfortunate limitation that we cannot investigate atmospheric and ionospheric effects and antenna angle-of-arrival effects (gain and polarization) on the attacks, it is the safest approach. Of course a real attacker need not be constrained by US law, and would not use a coaxial cable. A sufficiently large protected chamber that shields RF emissions, e.g., an anechoic chamber could allow more experimentation with antenna effects and multiple GPS sources. However, protected chambers also typically shield incoming signals (including the real GPS signal), making it difficult to come up with a safe yet realistic experiment. With the RF output connected to an antenna rather than directly to a device, our spoofing range for a receiver would be limited primarily by power output, line of sight, and knowledge of a target's location. If these conditions are satisfied, spoofing of receivers up to 100km away is feasible.

The PCSS is connected to an antenna mounted on the roof. Signals were synthesized in real time as the experiments were performed. We verified that each receiver could acquire satellite lock and compute a navigation solution when the PCSS simply passed-through the real GPS signal before proceeding with our testing.

[2]Blackbox fuzzing is impractical for GPS due to the amount of time it takes for the receiver to get navigation data (tens of seconds) and the inability to get feedback from GPS devices due to their closed nature.

5 Attacks

Our attacks view GPS receivers as a software system. Each layer of the system is roughly analogous to layers in the OSI model, with its own set of security vulnerabilities and associated defenses. In this section we detail our attacks. In the next section we address defenses. We investigated several attacks in the course of our research; due to space we only detail attacks that were successful against at least one receiver. Table 1 shows our overall results.

5.1 Data Layer Attacks

The below attacks are carried out by manipulating data in the navigation message of the GPS signal. Therefore, previous spoofers which could only change the offsets in the PRN are not capable of carrying out any of these higher level attacks. Further, vulnerabilities at this level are due to software bugs in the processing of the navigation message, making them an entirely different class of attack from previous spoofing.

Middle-of-the-Earth Attack Recall the ephemeris data contains sufficient information for a receiver to calculate each satellite's position. Part of the ephemeris data is the square root of the semi-major axis $\sqrt{A}$ of the satellite's orbit. In our attack, we set $\sqrt{A} = 0$. This is similar to telling the receiver that the satellite is in the middle of the Earth.

We performed this attack on all receivers in order to determine which would reject the bogus data. All receivers except the NetRS rejected the data. In the Trimble NetRS, the gpssd daemon caught an exception and died. We speculate that the exception was a division-by-zero error, though there is no way to verify this without source (the actual error seems to be triggered in the closed-source firmware). Let μ be the WGS-84 value of the Earth's gravitational constant. The GPS specification has the receiver compute the mean motion n_0 of the satellite (rad/sec) as [3]:

$$a = (\sqrt{A})^2 \quad n_0 = \sqrt{\tfrac{\mu}{a^3}}$$

The NetRS attempted to resolve the error by rebooting. Unfortunately, the NetRS appears to cache ephemeris data. Caching is quite common and is the reason a warm-boot of a GPS receiver takes less time to acquire lock. Since the ephemeris was cached, the receiver made the same faulty calculation, again erred, and the system entered an infinite reboot cycle. The receiver only recovered after a full hardware reset was manually performed.

The almanac data also contains the same parameter $\sqrt{A}$. We carried out the attack successfully against the NetRS, again with the same continual reboot behavior, using the almanac parameter $\sqrt{A} = 0$. This experiment shows that the NetRS does not appear to correlate the ephemeris data with the almanac data.

To the best of our knowledge, our attack is the first demonstration of a cyber-attack (a DoS) over RF against GPS. Overall, in our attack the attacker needs only transmit the bogus data until the receiver decodes the ephemeris, which typically takes 30 seconds. After the ephemeris is decoded, the receiver will enter a reboot cycle *even after the attack is stopped.* Thus, the attack achieves a similar goal to jamming: denying service, but unlike jamming, does not require continual broadcast. While DoS is not considered especially harmful in typical computer networks, availability of the GPS heartbeat is typically critical in applications that actually use

	uBlox	UAV	iFly	eXplorist	eTrex	NetRS	Arbiter
Vulnerable to Middle-of-the-Earth						√	
Vulnerable week #	√	√	√		√	√	√
Vulnerable to Date De-synchronize							√
Vulnerable OS			√	√		√	
Spoofable	√	√	√	√	√	√	√

Table 1: Successful attacks against receivers.

GPS. As the NetRS composes about 30% and 20% of the global CORS and NTRIP networks, respectively, the implications of this attack are widespread.

A demonstration of this attack on the NetRS receiver has been made available at `http://youtu.be/6K8dD2PCI6s`.

Vulnerable Week Number Date calculations in GPS receivers are done using the Z-count, which consists of a 10 bit Week Number (WN) and 9 bit Time of Week field.

In our attack, we first set the week number to be one past the current week. No other data was changed in the navigation message. When the ephemeris expired (the IODC and IODE changed), all receivers except the eXplorist accepted the new week number. We were then able to set the week number to any value in the 10 bit range.

Date De-synchronization Attack With only 10 bits in which to store the WN, rollovers occur about every 19.7 years. Rollover is understood and expected in GPS, as the first event took place over a decade ago in 1999. The handling of these rollover events is still left up to the manufacturer's discretion in the original GPS specification [3]. In the specification for the next-generation GPS, the WN field is planned to be extended to 13 bits, leading to less frequent epoch rollover of once every 157 years.

To see the extent to which we could alter the date on the GPS, we simulated rollover events. In the attack, rollover is simulated by alternating between high (all 10 bits set), low week numbers (1-5 bits set), and medium week numbers (8-9 bits set). [3] We also set the IODC and IODE to arbitrary (new) values, telling the receiver that the data issued is new and should be decoded. At this point, the GPS receiver should use an internal clock to decide whether it is reasonable for a rollover event to be occurring. In our Arbiter receiver, however, no internal clock comparison appears to be done, and the device increments the week epoch faithfully.

The Arbiter suffered permanent damage from this attack. The clock handles week rollover using an internal counter stored in non-volatile memory. This way, if the received week number reaches the maximum and then decrements, it is due to a rollover event and the week epoch is incremented. In normal operation, such behavior would indeed be correct, however, it is possible to broadcast a GPS signal with a false week number. By slowly alternating between high and low week numbers, the receivers perceived the GPS week epoch rollover was occurring, each time incrementing the base date by almost 20 years. The substation clock provides no way to decrement the week epoch. Multiple days without power, attempts to change the date through commands over the serial console, and reloading the firmware of the device proved unsuccessful for decrementing the year on the clock, rendering the device practically useless as a sub-microsecond accurate time source. Figure 5 shows the Arbiter's display when it acquires the unmodified GPS signal

[3]We found without using intermediate week numbers, the receiver could revert backwards rather than move forwards.

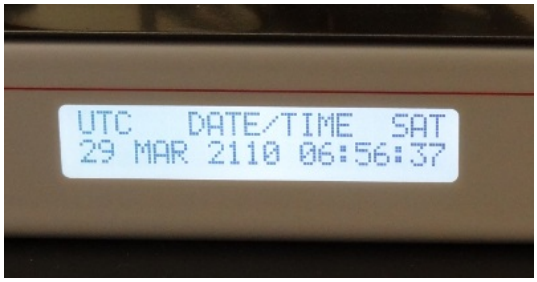

Figure 5: Arbiter PMU clock time after date-desync attack.

after the attack has ceased. This date de-synchronization attack depends on the manufacturer's handling of week rollover events, and will not be alleviated by increasing the length of the WN by 3 bits. In fact, increasing the time between week rollover events increases the severity of this attack. As our week rollover attack pushes time forward on the receiver by the length of the week epoch, increasing this time frame simply allows us to jump forward by even larger amounts of time.

5.2 The OS Layer

GPS receivers are typically treated as hardware devices, yet they often run a full OS stack. We identified the OS on three receivers: the Magellan, the iFly, and the NetRS. In all cases we were able to gain root access.

The NetRS The NetRS runs Linux, and can be networked via an Ethernet port. Network-facing services include an NTP server, an HTTP server, and an FTP server. While we would expect them to be on private networks, they also often appear on the Internet at large (e.g., by Googling identifying strings in the NetRS web interface). We discovered numerous security flaws.

First, the receiver's web configuration interface requires no password by default. Further, even if a user chooses to require login credentials, there is still an administrative user included by default with a password which is the same across all receivers.

By navigating to the Programmatic Interface, a malicious user can upload a configuration file to a Perl CGI program with a specially crafted filename. The NetRS then copies this file to a temporary location, however, it does this by calling the Linux `cp` command with an improperly sanitized version of the filename which was uploaded. This allows a carefully crafted filename to execute arbitrary commands on the system as the `www-data` user.

Further, the `www-data` user is given `sudo` privileges for some basic commands such as `cp`, `mount`, and `mv`, trivially allowing an attacker to gain root privileges on the machine by replacing the `/etc/sudoers` file with a version which gives the `www-data` access to run all commands with root privileges. Once on the system, an attacker could poison the GPS data to interfere with GPS measurements, or use the computer as a normal Linux host.

Second, Trimble uses a single master Linux image for all installations, called the "firmware" at their website. The current firmware has a relatively old Linux revision: 2.4.19. The image contains a copy of all programs, which allow an attacker to easily find additional vulnerabilities. Worse, the firmware has a default password for root in /etc/passwd with no public interface to change it. Thus, it appears likely that most receivers on the Internet share the same root password. Although we used a password cracker to attempt to find the password, we were not successful after running it for months on a modern CPU. This suggests that at least the password used in the NetRS is relatively strong.

Finally, we note the NetRS image did not have modern OS defenses such as ASLR enabled. The NetRS also did not appear to have any way to perform software updates other than installing a new firmware image, making patching onerous.

Windows System Both the Magellan and iFly receivers run on top of Windows CE 5.0. Both allow access to their file systems through USB or an SD card slot for the purpose of updating GPS maps and device firmware. By uploading executables through either of these two interfaces, we are able to run arbitrary code, including Windows Explorer and its associated tools. This could lead to malware or identification of new software vulnerabilities which could be exploited on the Data Layer.

This problem has been seen in mobile devices, such as smartphones, in the past. In general, manufacturers can solve this by requiring code to be signed in order to run or by restricting direct access to the file system on the device. While some users may see the ability to use their GPS receiver as a general purpose computer as a feature rather than a vulnerability, and indeed such communities exist for using GPS devices effectively as PDAs, the security ramifications of running untrusted code on devices intended to navigate and track people are serious.

5.3 GPS-Dependent Systems

GPS systems are used as an authoritative time source in many applications. Here we show the ramifications of the ability to spoof and damage GPS receivers to dependent systems. Our goal is not to provide an extensive list (which is likely very long), but to demonstrate that attacks on dependent systems are indeed practical.

Fixed-Width Year Attacks Recall that we can increment the date by approximately 20 years per ephemeris decoded by the receiver. The date de-synchronization attack gives us a black-box way to remotely (via RF) cause rollover in fixed-width date fields, similar to the Y2K problem.

For example, the Arbiter provides a time synchronization service for Windows XP. Windows XP dates must be in the range 1980-2099. Although the system will continue to operate after this range of times, some critical aspects of the system, such as file time-stamps used in FAT file systems will not work, as they are stored as unsigned 7 bit integers. In our experiments, we confirmed the date could be incremented at least up to 2110 (see Figure 5) and alter Windows XP dates.

A second example is the UNIX epoch roll-over will happen in 2038 for software that uses a 32-bit signed number for the date (e.g., `time_t` on Debian Squeeze). We experimented with 32-bit Debian Squeeze install, and found that the `time_t` and `struct timeval` data structures use 64-bits. However, the ext3 file system was vulnerable: after incrementing the date past 01/18/2038 22:14:14, rollover occurred in the time stamps for files, as well as the output of the `date` command. Therefore, applications that trust the `date` command, or perform critical operations based upon file system dates, may be vulnerable to attack.

Decoding takes about 30 seconds, which means in principle we can increment about 40 years per minute. The first 6-digit date is the year 100,000. So we would need to increment about 97,989 years to cause rollover, which would take about 1.7 days.

NTRIP The NetRS is often deployed as a reference station which not only receives GPS information, but also transmits correction data to other receivers. The CORS and NTRIP networks are designed to facilitate locating nearby correction data sources and transmitting of the correction data to clients. Since deliberate GPS signal degradation was disabled, a typical GPS receiver will experience a position error of around 10-15 meters [1]. NTRIP attempts to give centimeter-level accuracy, e.g., for autonomous driving of tractors or other vehicles.

Decimeter-level accuracy is possible by using differential GPS techniques. For example, several stationary NetRS's can be configured to use differential GPS to calculate accurate true position information. The NetRS then continues to calculate the current position and use measured position changes to estimate GPS errors, which can be broadcast as a correction solution for nearby devices. Variations are due to changes in local atmosphere, satellite ephemerides, etc. NTRIP sources broadcast the correction data over the NTRIP protocol, which can run on top of IP, EDGE, or other network protocols. The protocol is designed to also help clients find the closest NTRIP source. The closest source is desirable because relative error sources, e.g., atmospheric conditions, are most likely to be similar. The NetRS and NTRIP are also capable of supplying Real Time Kinematic data, which provide even more accurate positioning information for nearby receivers.

We set up a NTRIP testbed consisting of the NetRS, which served as the correction data server (called a Caster in NTRIP), and several receivers. We confirmed that our attacks flowed through to clients connecting to our server, causing them in turn to calculate erroneous navigation solutions. We note that joining the global NTRIP network only requires filling out an appropriate form. [4] We did not test this as the network operators request a long-term commitment to provide data.

NTP Modern computers use NTP extensively time synchronization, with NTP clients packaged by default on nearly all modern operating systems. In turn, this time is used for many other purposes including cryptographic communication, shared file systems, and even data for the Health Insurance Portability and Accountability Act (HIPAA) which requires accurate timestamps on medical records.

NTP relies on computers connected directly to high precision clocks. In general, NTP servers are classified by their Strata, with a Stratum-1 server being directly connected to a clock, a Stratum-2 server connecting directly to a Stratum-1 server, and so on. GPS is commonly used as a Stratum-1 time source.

Our analysis showed that of the Stratum-1 servers in the public NTP.org pool, at least 148 servers [5] used GPS for time synchronization, or at least 65% of the pool.

NTP works by retrieving time estimates as well as error intervals ($t \pm \lambda$) from some client-specified list of servers. A time interval on which at least half of the queried servers agree is then selected, and the actual time reported is then based on a clock in this range. This means that directly spoofing the time for a client connected to multiple Stratum-1 servers could be done only if an attacker had

[4] `http://igs.bkg.bund.de/ntrip/registerprovider`

[5] Some NTP servers do not list the devices they use for time synchronization, so only a lower bound may be obtained.

the capability to spoof at least half of the servers used by a client at the same time. Although this may be feasible for a dedicated organization, it can be difficult especially if clients specify numerous, geographically separated NTP servers.

However, given an NTP client which normally reports a time interval $I = [t - \lambda, t + \lambda]$, an attacker who can modify only one NTP server (say, by spoofing the GPS clock used by the server) may change the time reported to the client to a subset of I or prevent the client from selecting a clock by forcing that there does not exist an interval on which a majority of servers agree. If a client queries only a small number of servers, which was typical, such attacks are readily done. This would lead to growing inaccuracies in the client's clock over time, though the inaccuracies would not be entirely under attacker control. [7]

Monitoring an NTP client on an author's computers, (see Figure 6) we noted that interval widths in typical operation were typically from hundreds milliseconds to one second.

This amount is small for most consumer purposes, but considering NTP accuracy should be on the order of 10ms over a WAN, pushing time towards an extreme end of the interval could have a large impact. For example, industrial automation, SCADA systems, and operations management commonly need sub-second timing for proper operation, and commonly use NTP to accomplish this.[14]

5.4 Pseudorange Attack

We have also independently verified the results of previous work in spoofing [4]. In this attack we estimated the received GPS signals parameters, and then synthesized new signals at different relative PRN code offsets.

Recall a receiver calculates its position in space-time via trilateration. If we modify one of the satellite's PRN offsets, but not others, then we affect one dimension of the navigation solution. If we shift all PRNs by an offset of the same amount, we only affect the time dimension of the navigation solution. More generally, by varying the PRN phase for particular satellites in a concerted way, we can make a receiver's navigation solution move in space-time.

We created two categories of spoofing attacks: one that manipulated position and one for time. The time and position spoofing can be carried out independently, or in combination. For position, we created two spoofed modes. First, we spoofed receivers going in a particular direction at a particular velocity. Second, we spoofed receivers going in a circle, with a configurable diameter, speed, and direction parameter. For the time spoofing, we have developed three attacks that varied the rate at which time is changed as either linear, exponentially, or logarithmic. All receivers accepted the spoofed data for all attacks.

6 Defenses

Previous work has proposed several hardware defenses to detect spoofers, e.g., [4, 6, 13, 20]. Example solutions include using multiple antennas to detect the spoofer and adding cryptographic authentication to the civilian signal. These are good long-term suggestions to improving GPS security. However, current proposals require hardware or other significant modifications, e.g., by adding new antennas or adding a hardware decryption unit (called the SAASM in military receivers), making them inadequate in the short term. Papadimitratos et al. [12], discuss a detection method based on Doppler Shift observations during satellite takeover. This technique can detect some spoofing attacks on certain high-end receivers without any hardware modification. In this section we describe device-specific software security recommendations, as well as propose electronic GPS attack detection and whitening systems.

As this defense system functions at the data-level and higher, it functions only to protect against attacks at a similarly high level. Protecting against traditional spoofing attacks would still require the capability to examine the physical-layer of the signal, which necessitates some form of hardware device as detailed in previous work.

6.1 Device-Specific Recommendations

Data-Level Protection One observation is that sanity checking on input is spotty in receivers. The NetRS software notices an error and attempts to reboot the receiver during our attack. However, it does not recognize that the cache should be cleared as well, leading to a reboot loop. The date de-desynchronization attack is a failure to check for consistency of the GPS week number with what could be known from a cheap internal clock. The fact that the damage is permanent on the Arbiter seems to be a software flaw: Arbiter provides a serial configuration console which should allow changing the date, but it does not work correctly to reset the date.

It is also interesting to note the more expensive receivers tended to have more problems than cheaper receivers in our experience. One possible explanation would be that more expensive receivers have more functionality and perform more complicated processing of the GPS signal, and thus have an increased risk to programmer error.

OS-Level Defenses Our experiments show that GPS users and manufacturers view receivers as hardware devices, not computers. Nonetheless, they contain software vulnerabilities. While these attacks may come as no surprise for those well acquainted with software security, they appear often ignored by those in charge of deploying GPS receivers for widespread usage.

At first glance our remote attack against the NetRS may seem naïve: one may not expect most GPS receivers to be networked, least of all accessible through the internet. However, simple Google searches revealed over a dozen NetRS receivers whose web interfaces were accessible remotely (despite the web server of the NetRS disallowing indexing on all pages by default). Further, because these are CORS receivers, they are almost always on some type of network in order to send their correction data to other devices. In fact, we were able to find publicly accessible websites for other CORS receivers such as the Trimble NetR5 and the Leica GRX1200+, showing that it is not uncommon for CORS receivers to be accessible from the public internet. Apart from simply using access to these devices to disrupt GPS measurements, there are many other attack scenarios. For example, several receivers which are web accessible reported internal NAT IP addresses, meaning a compromise of the GPS receiver could function as a launching point for attacks on the internal network.

The issue of running untrusted code on GPS receivers is more of a symptom rather than a problem itself. The requirement of physical access makes this vulnerability a very unlikely attack vector, but is indicative of a larger attack surface. First, because these receivers commonly run very large software stacks with full blown operating systems and due to the specialized nature of these devices, much of the included software does nothing but contribute to the already large number of programs in which bugs may be found. Second, the manufacturers of these devices did not have security in mind when developing these systems. Even if loading code onto a physical device is not a likely vector for widespread attacks, code signing on embedded systems at some level should be a common practice,

as untrusted code posing as maps, games, or firmware updates may easily break a user's device either maliciously or accidentally.

One immediate best practice would be for GPS receiver manufacturers to build and deploy automated software update mechanisms. At present, users typically must go to the manufacturers home page, download the update, and then transfer it to the receiver. Other recommendations include receivers white-listing programs that can run, and implementing modern OS defenses such as ASLR and DEP.

GPS Dependent Systems The NTP protocol uses a filter to limit the damage of malicious navigation information. This is a promising sign for protocols which rely upon GPS *assuming that multiple independent sources are used.* We speculate, however, that many corporate and government environments will only use a single internal GPS time source as a time server in order to limit outside connectivity in their firewall, or as a matter of convenience. We recommend users re-evaluate their risk levels with respect to our findings: using only a single receiver as a time source leaves a user vulnerable to spoofing and other attacks, while using external time sources via NTP likely require appropriate holes in the firewall.

Reference networks, such as CORS and NTRIP, broadcast correction data and are used for a large number of applications, and thus should also adopt defenses that utilize multiple independent sources. The NTRIP network consists of three types of entities: NTRIP sources (the GPS receivers), NTRIP servers (for broadcasting correction data), and NTRIP clients (which receive the correction data) [1]. NTRIP is used widely in agriculture, navigation, and surveying. Unfortunately, the NTRIP protocol does not specify how aggregation among multiple sources is to be performed. One recommendation is to specifically address counter-measures to detect or filter out anomalous readings.

6.2 EGADS

Our attacks show that serious damage can be done to life and safety-critical applications using only a few thousand dollars worth of hardware. Since replacing or updating existing equipment may be difficult, we propose deploying Electronic GPS Attack Detection Systems (EGADS). An EGADS is similar in spirit to a network or host IDS system, but designed to detect GPS attacks. To the best of our knowledge, we are the first to propose using an IDS style system to detect GPS spoofing attacks [6].

Our EGADS design has a rule-based and anomaly-based component. The rule-based component detects known bad values, e.g., $\sqrt{A} = 0$, while the anomaly-based engine detects deviations given known good almanac data. We have implemented EGADS on top of a GlobalSat ND-100S USB GPS Receiver module based off of the SiRF-III GPS chipset. This provides an output data stream consisting of normal positioning values as well as almanac and ephemeris data from the satellites.

The rule-based engine performs a pattern match on GPS parameters. To detect anomalies, we modified `gpsd` to check the ephemeris data reported by a receiver against the freely available almanac data provided by the United States Coast Guard. In Figure 7, we show typical discrepancies over a two week time span, measured every two hours, between historical ephemeris values for $\sqrt{A}$ (retrieved from the NASA Crustal Dynamics Data Information System) and the reference almanac used in EGADS for each particular day. Because these values are relatively static (though not all remain as static as the $\sqrt{A}$ parameter), we find that the straight forward approach to detection is effective. We combine these recordings with our own measurements, and double [7] the largest discrepancy historically seen for each ephemeris field and for all satellites we observed, giving us a range outside of which we should never fall. If the receiver does report values outside of this range, EGADS will issue a warning to the user.

Assisted GPS (AGPS) techniques also provide a fallback for accurate navigation in areas where the GPS navigation message is being disrupted. AGPS uses a set of ephemerides and almanac data from external sources to acquire a GPS fix more quickly. An AGPS receiver could be modified (as standard AGPS devices only use external data to establish a fix) to ignore the Navigation Message completely, and instead rely on data from a secure side channel to continue operation.

Although this technique does not detect GPS spoofing of position and timing data from the C/A code, it will warn of attempts to crash receivers by creating bogus data in the GPS navigation message. Our system can also detect attempted date de-synchronization attacks easily, as it does not trust the GPS more than the internal computer clock. Further, this method detects things such as fuzzing of ephemeris or almanac fields, and to some extent a replay of recorded GPS data, all within seconds of broadcast.

As future work, we propose GPS whitening systems. Whitening takes in a potentially anomalous or malicious signal, and retransmits a known good signal. For example, we could augment the PCSS to remove attacks from the GPS stream and rebroadcast a clean signal to nearby receivers. This would improve upon the device in [5] by protecting against data-level attacks.

7 Conclusion

The intricate nature of today's GPS devices has created a large attack surface. Previous approaches have treated GPS security as an issue of hardware and signal analysis, but many traditional software security lessons have yet to be learned by GPS manufacturers. In this paper, we introduced novel attacks on GPS devices at the data level, many of which have serious ramifications to safety systems. Our attacks required hardware that cost only about $2,500, which is about the same as a laptop. We have shown our attacks are successful against consumer and professional receivers. We also propose defenses such as hardening GPS software against RF and network attacks, as well as an attack detection system. Until GPS is secured, life and safety-critical applications that depend upon it are likely vulnerable to attack.

8 Acknowledgements

We would like to thank Srdjan Capkun and the anonymous reviewers for their helpful feedback.

References

[1] RTCM Special Committee No. 104. *RTCM Standard 10410.1: Networked Transport of RTCM via Internet Protocol (NTRIP) Version 2.0.* Radio Technical Commission for Maritime Services, June 2008.

[2] James Carroll, Karen Van Dyke, John Kraemer, and Charles Rodgers. Vulnerability Assessment of the Transportation Infrastructure Relying on the Global Positioning System. In *ION National Technical Meeting*, 2001.

[6] GAARDIAN is a proposed system for detecting interference, such as from multipath errors, weather conditions, or jamming, and is not designed for detecting spoofing attacks [8].

[7] We chose to double our difference to ensure no false positives were reported, though this could be adjusted easily for different usage scenarios/attacker models.

[3] ARINC Research Corporation. Navstar GPS space segment/navigation user interfaces (the interface control document). Technical Report IRN-200E-004, 2010.

[4] Todd E Humphreys, Brent M Ledvina, Mark L Psiaki, Brady W O Hanlon, and Paul M Kintner. Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *Proceedings of the Institute of Navigation GNSS (ION GNSS 2008)*, 2008.

[5] Brent M Ledvina, William J Bencze, Bryan Galusha, and Isaac Miller. An In-Line Anti-Spoofing Device for Legacy Civil GPS Receivers. In *Proceedings of the Institute of Navigation International Technical Meeting (ION ITM 2009)*, January 2009.

[6] Sherman Lo, David Lorenzo, Per Enge, Dennis Akos, and Paul Bradley. Signal authentication: A secure civil GNSS for today. *Inside GNSS*, October 2009.

[7] D. Mills, Ed. J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: protocol and algorithm specification. Request for Comments RFC 5905, 2010.

[8] Carl Milnder, Washington Ochieng, Wolfgang Schuster, Marco Porretta, and Charles Curry. A regional space segment health monitor for local gps integrity monitoring. *Journal of Navigation*, pages 657–671, 2011.

[9] Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2nd edition, 2006.

[10] National Oceanic and Atmospheric Administration. Continually operating reference station (CORS). `http://www.ngs.noaa.gov/CORS/`.

[11] California Department of Corrections and Rehabilitiation. Electronic monitoring unit. `http://www.cdcr.ca.gov/Parole/Electronic_Monitoring_Unit/index.html`.

[12] P. Papadimitratos and A. Jovanovic. GNSS-based Positioning: Attacks and Countermeasures. In *Military Communications Conference*, 2008.

[13] Mark L. Psiaki, Brady W. O'Hanlon, Jahshan A. Bhatti, Daniel P. Shepard, and Todd E. Humphreys. Civilian GPS Spoofing Detection based on Dual-Receiver Correlation of Military Signals. In *Proceedings of the Institute of Navigation GNSS (ION GNSS 2011)*, September 2011.

[14] Schneider Electric. PowerLogic SCADA: Power Monitoring and Control Software, 2008.

[15] Logan Scott. Anti-spoofing and authenticated signal architectures for civilian navigation systems. In *Proceedings of the Institute of Navigation GPS/GNSS 2003*, pages 1543–1552, 2003.

[16] Spirent. Using receiver NMEA data in a simulator test scenario. Technical Report DAN007-TM Issue 1-00, Spirent, unmarked year.

[17] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. On the requirements for successful GPS spoofing attacks. *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 22:75–85, 2011.

[18] A J Van Dierendonck. *Global Positioning System: Theory and Applications*, chapter 8: GPS Receivers, pages 329–407. American Institute of Aeronautics and Astronautics, Washington, D.C., 1996.

[19] Jon S Warner and Roger G Johnston. A Simple Demonstration that the Global Positioning System (GPS) is Vulnerable to Spoofing. *Journal of Security Administration*, 2003.

[20] Kyle Wesson, Daniel Shepard, and Todd Humphreys. Straight talk on anti-spoofing. In *GPS World*, January 2012.

APPENDIX

Although the main body of the paper reflects our overall experimental results, the raw data collected can be seen in Figures 6 and 7.

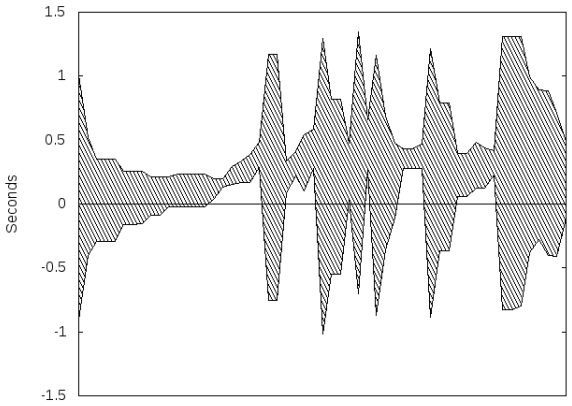

Figure 6: NTP time interval observed over a 3 hour period, polling from `time.windows.com`, `time.ubuntu.com`, and `time.nist.gov`.

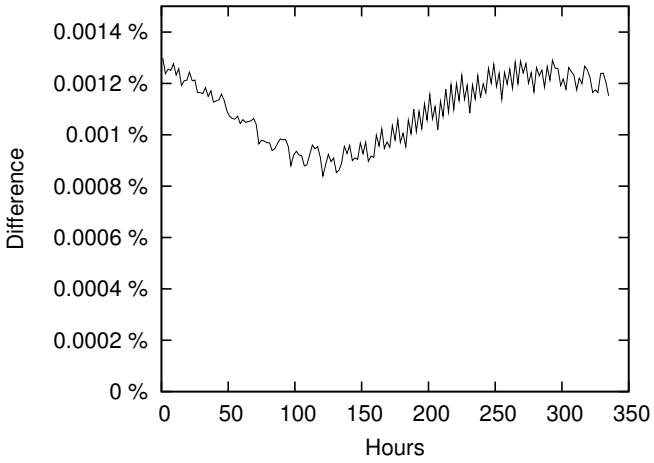

Figure 7: The level of difference of the $\sqrt{A}$ value of the ephemeris against that of our reference almanac over a two week period.