Cyber Attack Vulnerabilities Analysis for Unmanned Aerial Vehicles

Alan Kim, * Brandon Wampler, † James Goppert, ‡ and Inseok Hwang §
Purdue University, West Lafayette, Indiana, 47907, United States
Hal Aldridge ¶
Sypris Electronics, Tampa, Florida, 33612, United States

As the technological capabilities of automated systems have increased, the use of unmanned aerial vehicles (UAVs) for traditionally exhausting and dangerous manned missions has become more feasible. The United States Army, Air Force, and Navy have released plans for the increased use of UAVs, but have only recently shown interest in the cyber security aspect of UAVs. As a result, current autopilot systems were not built with cyber security considerations taken into account, and are thus vulnerable to cyber attack. Since UAVs rely heavily on their on-board autopilots to function, it is important to develop an autopilot system that is robust to possible cyber attacks. In order to develop a cyber-secure autopilot architecture, we have run a study on potential cyber threats and vulnerabilities of the current autopilot systems. This study involved a literature review on general cyber attack methods and on networked systems, which we used to identify the possible threats and vulnerabilities of the current autopilot system. We then studied the identified threats and vulnerabilities in order to analyze the post-attack behavior of the autopilot system through simulation. The uses of UAVs are increasing in many applications other than the traditional military use. We describe several example scenarios involving cyber attacks that demonstrate the vulnerabilities of current autopilot systems.

*Graduate Student, Aeronautics and Astronautics, awkim@purdue.edu, AIAA Student Member.
†Graduate Student, Aeronautics and Astronautics, bwampler@purdue.edu, AIAA Student Member.
‡Graduate Student, Aeronautics and Astronautics, jgoppert@purdue.edu, AIAA Student Member.
§Associate Professor, Aeronautics and Astronautics, ihwang@purdue.edu, AIAA Senior Member.
¶Director of Engineering, Sypris Electronics, Hal.Aldridge@sypris.com

I. Introduction

I.A. Background

The use of unmanned aerial vehicles (UAVs) has been limited to military use for the past decade. Some of the missions UAVs have been used for in that context include:

- Surveillance
- Reconnaissance
- Tracking
- Combat
- Support

Research conducted by Frost and Sullivan between 2004 and 2008 shows that the number of UAVs deployed globally on operations has increased from around 1,000 to 5,000 systems. [1] The growth in the use of UAVs will continue as the technology improves, leading to cheaper and more capable unmanned systems. Some of the non-military uses of UAVs are highlighted in Figure 1. As the number of UAVs in use increases, the potential for and interest in cyber attacks on the UAVs also increases. Some of the general unmanned systems problems have already been identified using the assumption that the unmanned system is a remote node of a network, as in [2], but there has not been any research focusing on UAV autopilot vulnerabilities specifically.

The interest in UAV cyber security has been raised greatly after the Predator UAV video stream hijacking incident in 2009, where militants used cheap, off-the-self equipment to stream video feeds from a UAV. With greater funding and skills, the possible damage due to a cyber attack is a major concern. The U.S. Army, Air Force, and Department of Defense have all shown clear interest in defending against cyber attacks on deployed UAVs. In depth research into cyber attack threats and vulnerability identification on these systems is therefore required. [3–5]

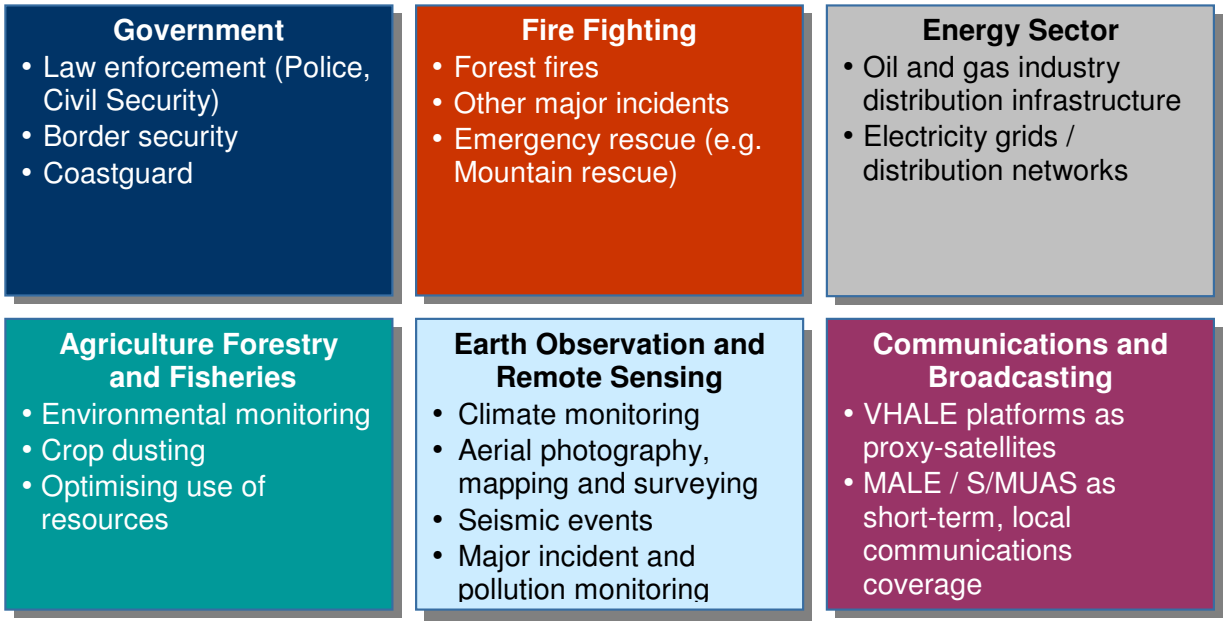


Figure 1: List of future uses for UAVs [1]

I.B. Scope

Our ultimate goal is to develop a cyber secure autopilot architecture for UAVs. For this initial study, we focused on:

1. Identification of the potential threats and vulnerabilities of current autopilot system.
2. Analysis of post-attack behavior of the UAV.

I.C. Assumptions

For this research we assume that it is possible to corrupt data within the data flow of the autopilot system through methods such as as buffer overflow attacks and other cyber attacks. [6] We do not consider the method of attack, but rather we look at the effect that maliciously corrupted data has on the UAV and the damage that results.

II. Study Methodology

II.A. Threats and Vulnerability Identification

First, a study on the current UAV systems, including components such as ground stations and satellites, was carried out in order to verify the data flow in and out of the UAV itself. Then, using our knowledge of the current autopilot system, we hypothesized ways of corrupting data in the autopilot data flow path. Literature reviews were carried out in order to gauge the possibility of the hypothesized attack methods.

II.B. Post-Attack Behavior Analysis

Using a high fidelity aircraft model, we carried out extensive numerical analysis in order to study the post-attack behavior of the UAV to specific cyber attacks. Sensitivity studies were also performed for each of the different cyber attack scenarios in order to determine the most effective attacks. The aircraft model and numerical analysis method will be covered in detail.

III. Statement of the Problem

The general autopilot system structure has not changed since it was introduced for manned aerial vehicles. Because the autopilot system was originally developed for manned systems, cyber security was not a design priority. This makes the current UAV autopilot systems vulnerable to malicious cyber attacks. [7] Our problem is to redesign the autopilot system to be robust to potential cyber attacks. In order to do this, we will identify the threats to and vulnerabilities of the current UAV autopilot system in the context of cyber attacks and analyze the post-attack behavior of the UAV.

IV. Current and Future UAV Systems

UAVs have been used by the military for over a decade with huge success. Currently, the military is changing their infrastructure to move towards more network-centric warfare, where all of the components of the military are interconnected through sophisticated networks. [8] This will provide fast communication and constant environmental and asset awareness for the entire military. Some UAV systems, such as the Global Hawk, already employ this type of infrastructure, as seen in Figure 2. It is important to notice that all the components of the system are interconnected, and an attack on one component can cause a propagation of failures throughout the whole system. In this project, we focus only on the cyber security of a UAV autopilot, and not on the larger network. The possibility of a UAV cyber attack causing failures in other network components should be noted nonetheless. As the military moves further into network-centric infrastructures and as civilian applications follow, the potential damage of a cyber attack increases.

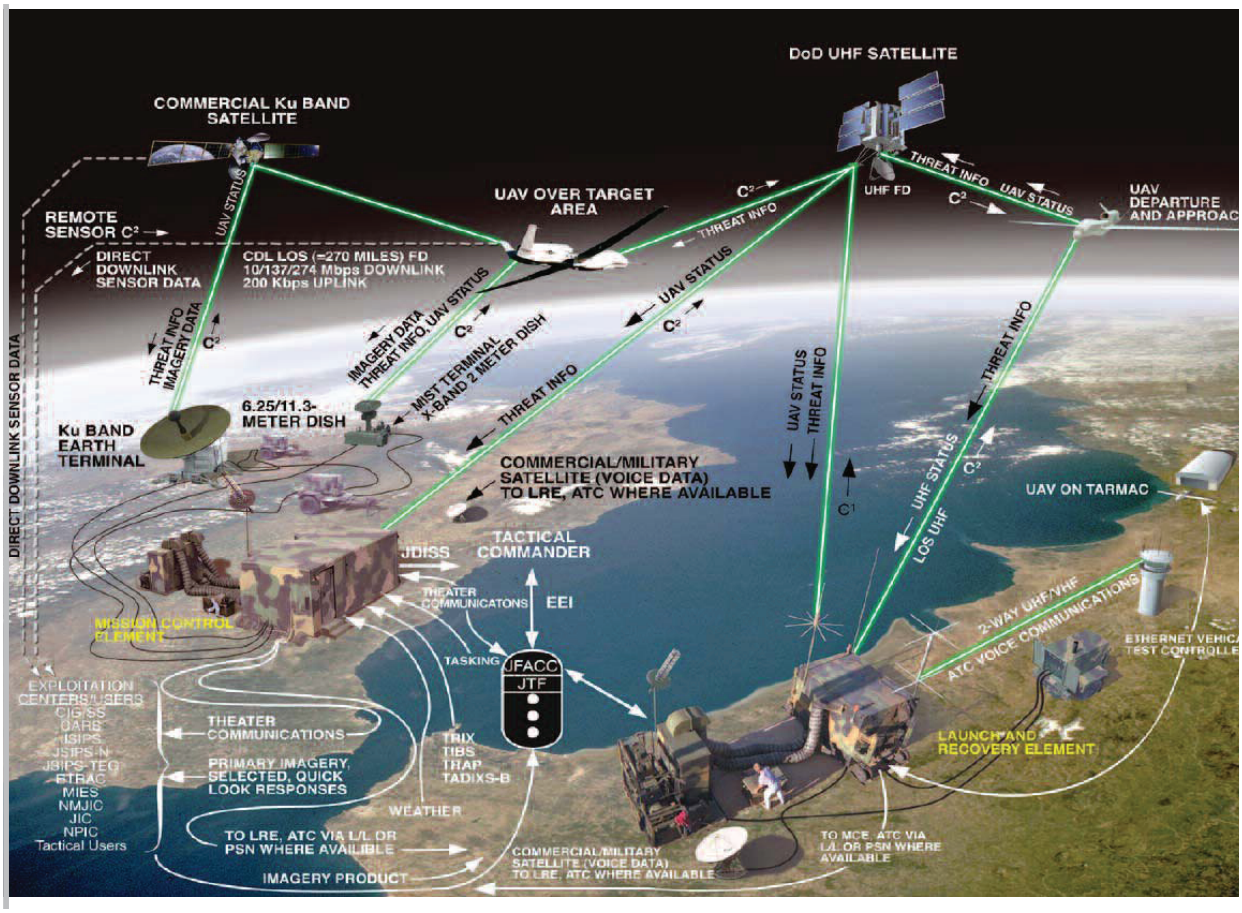


Figure 2: Global Hawk UAV Systems Architecture [9]

V. Current Autopilot Architecture

The common components found in UAV autopilots are:

- **Main Program/Processor**: Responsible for processing sensor data and the implementation of the control of the UAV.
- **Magnetometer**: Used for measuring direction.
- **GPS**: Used to determine the global position.
- **Airspeed/Altimeter**: Used to measure air speed and altitude.
- **UAV Wireless Communication**: Responsible for communicating with the ground station.
- **Power System**: Responsible for providing power to the entire UAV.
- **Inertial Measurement Unit**: Used to measure the movement of the UAV.
- **Boot Loader Reset Switch**: Used to load programs into the main program board.
- **Actuators**: Receives commands from the main processing board and moves the control surfaces.
- **Manual Flight Control**: Overrides the autopilot and gives control of the UAV control surfaces to the ground station.

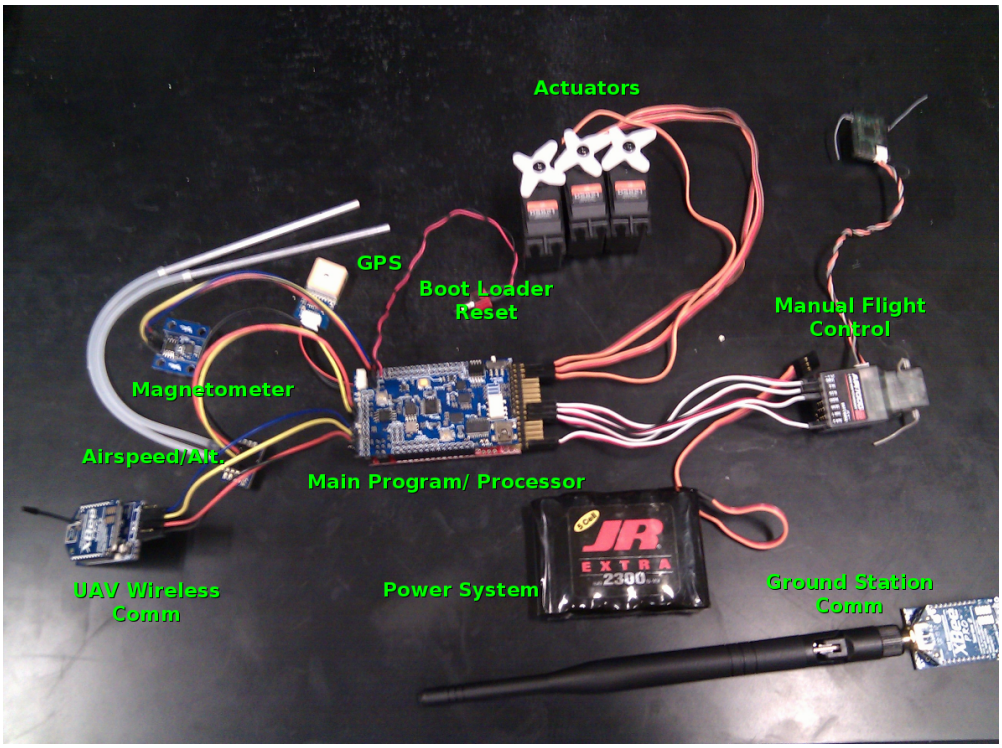


Figure 3: UAV autopilot components used at Purdue Hybrid Systems Lab (ArduPilotMega)

Figure 3 shows the components for the UAV autopilot used in our lab, the Purdue Hybrid Systems Lab. Although the specific equipment shown in Figure 3 is used for a small low cost UAV, it shares the same components found in larger and more expensive UAVs. For example, more expensive UAVs might have more powerful processing units or satellite communication links.

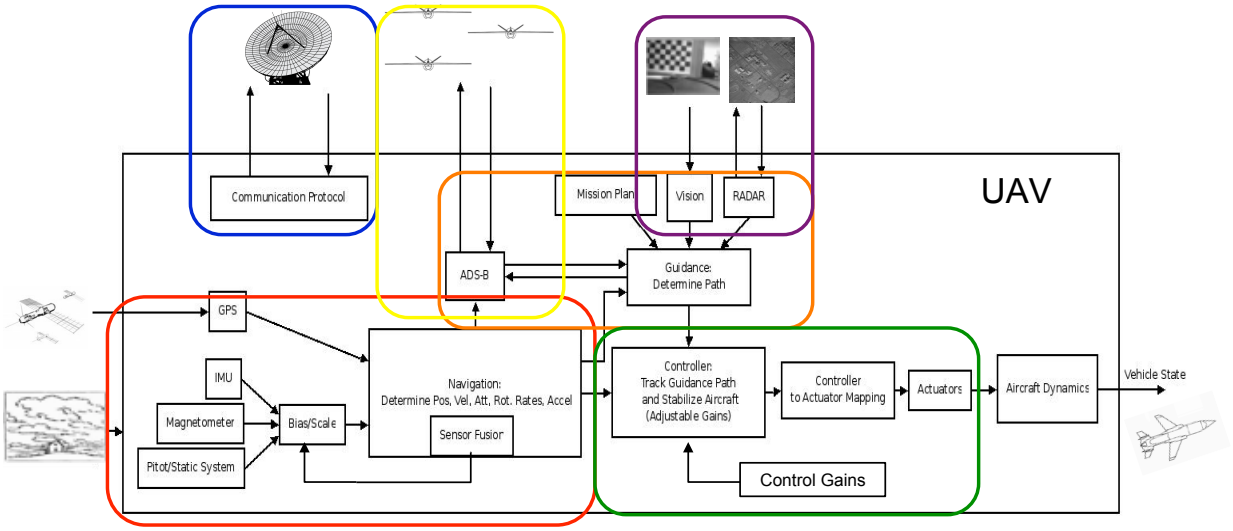


Figure 4: UAV autopilot architecture data flow diagram

Figure 4 shows the UAV autopilot components in a data flow diagram. In this diagram the components of the autopilot can be divided into three major parts which make up the GNC:

- **Guidance**: Determines a path based on the UAV's state, waypoints, mission objective, avoidance maneuvers, target tracking, etc (e.g., trajectory generation). Marked in orange in Figure 4.
- **Navigation**: Determines the UAV's state using sensory data. Marked in red in Figure 4.
- **Control**: Keeps the UAV in a safe, stable state in the presence of disturbances and steers the UAV to the target based on information from the Guidance and Navigation. Marked in green in Figure 4.

In Figure 4, the component marked in blue represents the communication in and out of the UAV, the component marked in yellow represents the new ADS-B communication, and the component marked in purple represents the sensors that can aid the guidance of the UAV.

VI. Threat and Vulnerability Identification

VI.A. General Attack Possibilities

We have determined, through studying the data flow in the autopilot, several general cyber attack feasibilities. These attacks have been categorized into three groups:

- **Hardware Attack**: Attacker has access to the UAV autopilot components directly.
- **Wireless Attack**: Attacker carries out the attacks through one of the wireless communication channels.
- **Sensor Spoofing**: Attacker passes false data through the on-board sensors of the UAV autopilot.

VI.A.1. Hardware Attack

Hardware attacks can occur whenever an attacker has direct access to any of the UAV autopilot components. An attacker can then corrupt the data stored on-board the autopilot or install extra components that can corrupt the data flow. These types of attacks can be carried out during the maintenance and storage of the UAV or during the manufacturing and delivery. An attacker can link directly to the UAV autopilot and damage it or reprogram it if he has the means or replace or add components which will give him control over the UAV and/or the tactical data collected. Hardware attacks can affect the survivability of the UAV, compromise control of the UAV, and compromise the tactical data collected by the UAV.

VI.A.2. Wireless Attack

Wireless attacks can occur if an attacker uses the wireless communication channels to alter data on-board the UAV autopilot. The worst case scenario for this attack is if an attacker is able to break the encryption of the communication channel. Once this occurs, an attacker can gain full control of the UAV if the communication protocol is known. Another possibility is an attack such as a buffer overflow that corrupts some data on-board or initiates some event. The most significant danger of wireless attacks is the fact that an attacker can carry out the attacks from afar while the UAV is being operated.

VI.A.3. Sensor Spoofing

Sensor spoofing attacks are directed towards on-board sensors that depend on the outside environment. Examples of such sensors are the GPS receivers, vision, radar, sonar, lidar, and IR sensors. An attacker can send false data through the GPS channels, or blind any of the vision sensors. The UAV autopilot relies heavily on sensor data for Guidance and Navigation, so corrupted sensor data can be very dangerous.

VI.B. Specific Attack Scenarios

We have further identified specific attack scenarios which show the vulnerabilities of the current autopilot system. Looking at these threats from the cybersecurity perspective, it was natural to further categorize the attacks into the following two types:

- **Control System Security**: Attacks that attempt to prevent the hardware/CPU from behaving as programmed. Some examples of this type of attacks include buffer overflow exploits through some input device, forced system resets to load malicious code, and hardware changes or additions to the system.
- **Application Logic Security**: Attacks that use malicious manipulation of the sensors or the environment, providing false data to the control system. In this case, the control system behaves as programmed without fault, but some or all inputs to the system are corrupted. Some examples of this type of attacks include sensory data manipulation, vehicle/system component state data manipulation, navigational data manipulation, and command and control (C^2) data manipulation. Figure 5 shows the most likely type of vulnerabilities for each component of the UAV autopilot.

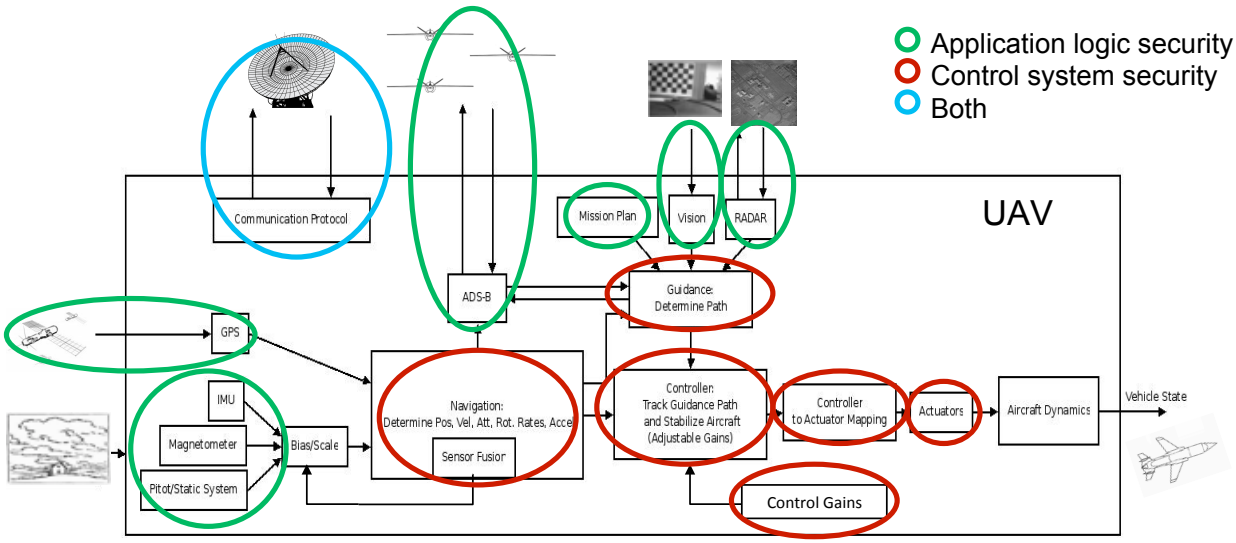


Figure 5: Vulnerabilities of the current UAV autopilot.

VI.B.1. Gain Scheduling

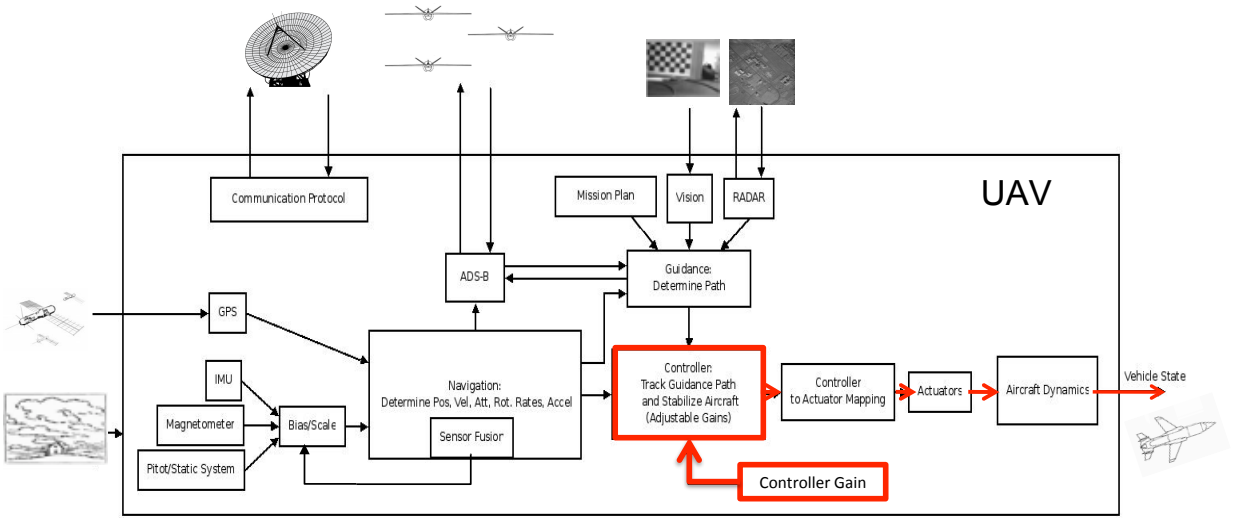


Figure 6: Flow of corruption due to gain scheduling attack.

Figure 6 shows the components that can be affected by the gain scheduling attack. This attack is an example of a control system security vulnerability. Gain scheduling is often used to control non-linear systems. For a example, a UAV will need different gains for control depending on the state of the UAV (mass, altitude, speed, flaps down, etc). A UAV will have different dynamical properties depending on its state and will require gains matched to each state in order to control the vehicle properly. Gain scheduling is also used in hybrid systems. In hybrid systems, a system is assumed to have multiple modes of operation, and the modes can change at any given time following some rules. In the case of a UAV, for example, there might be different modes corresponding to take off, landing, and cruising. Each of these modes will have different gains for controlling the vehicle.

The control gains are often pre-computed and trusted, and they are coded into the on-board autopilots. Without strict monitoring of the software, an override of the these gains could very well go undetected. Changes to the gains or gain scheduling logic could cause decreased performance in the autopilot or dangerous instability in a UAV. Additionally, even if all of the individual modes in a hybrid system are stable, some switching sequence could result in the system becoming unstable. This possibility could be used against the vehicle by an attacker. An attacker could also force infinite switching between gains, which will cause loss of control over a UAV.

Some of the possible attack methods are sensor spoofing to cause mode confusion, overriding gains through hacking, and causing denial of service between the controller gain block and the UAV controller block by overloading the on-board processor.

VI.B.2. Actuator/Sensor

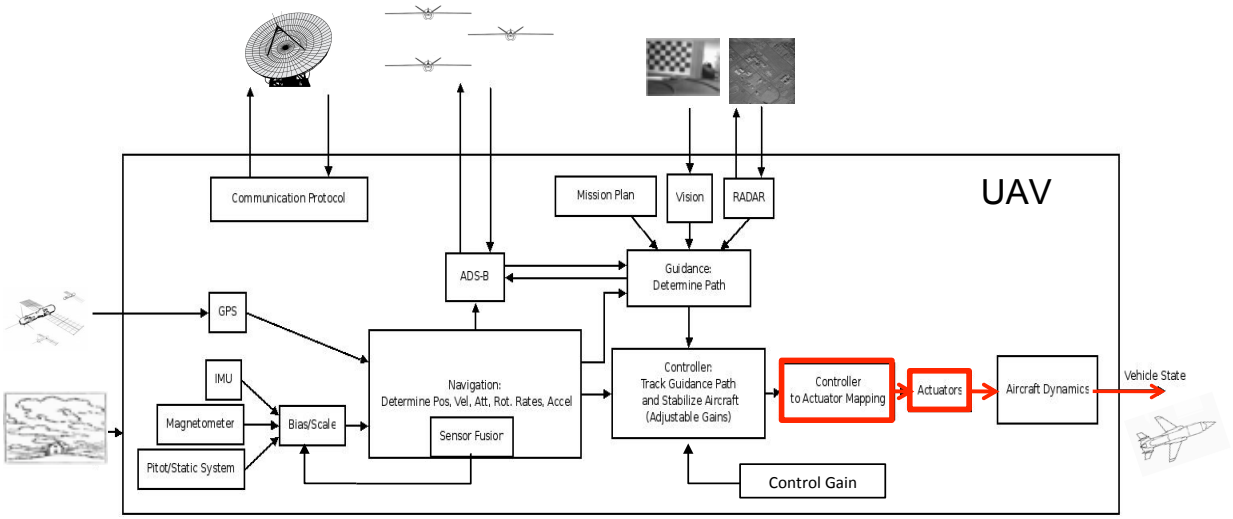


Figure 7: Flow of corruption due to actuator/sensor attack.

Figure 7 shows the components that can be affected by the actuator/sensor attack. This attack is an example of a control system security vulnerability. One of the easiest ways for corrupted data to affect the UAV control will be by affecting the control surface actuators and accompanying sensors. The UAV autopilot sends out commands to the actuators in order to modify the control surface state, which will modify the evolution of the vehicle state. If the commands going to the actuators were somehow overridden, the result would be the loss of control of the vehicle. Modifying the measurements of the actuator sensors in the systems that use them will also affect the state of the actuator and, thus, the state of the vehicle. This attack can result in loss of control of the UAV and instability in the UAV.

Some of the possible attack methods are overriding actuator state estimates, actuator mappings, or sensor readings; false data injection; and denial of service between the actuator and the controller interface by overloading the on-board processor. These attacks can be detected using a fault detection approach, but the algorithms currently being used are not robust enough or are too computationally costly. [10]

VI.B.3. GPS

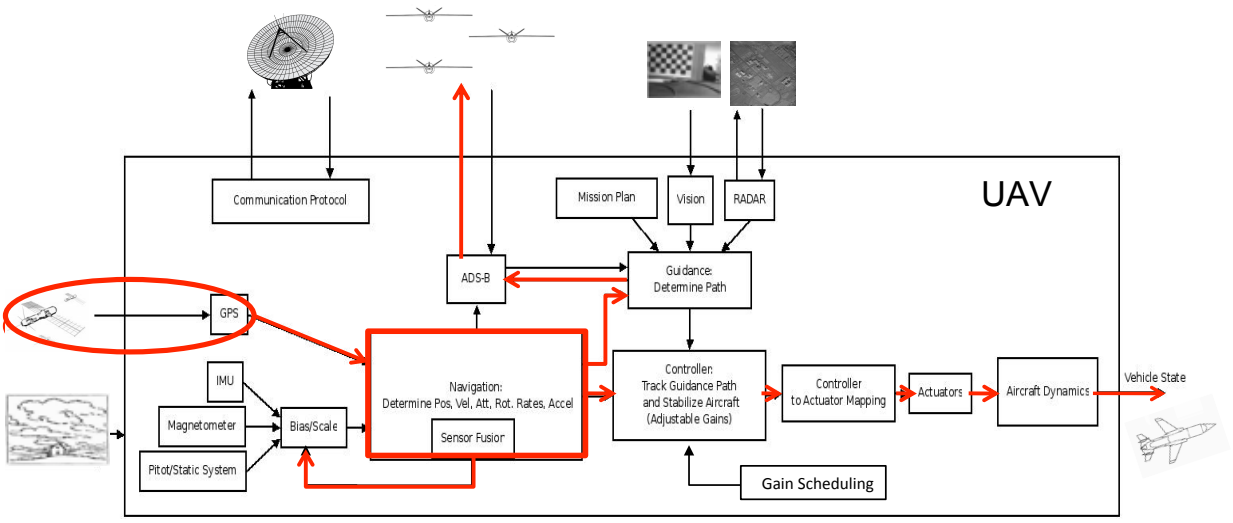


Figure 8: Flow of corruption due to GPS attack.

Figure 8 shows the components that can be affected by the GPS attack. This attack is an example of an application logic security vulnerability. Today, most UAV systems rely heavily on GPS data to locate themselves, the ground station, and their targets. The data received through the GPS sensors can be spoofed, which results in a false estimate of the UAV position in the on-board navigation system. If the UAV is fully automated, the on-board guidance system will then lead the UAV to a false target location or ground station if returning home. This type of attack will result in failed missions and possible loss of assets.

VI.B.4. Automatic Dependent Surveillance - Broadcast (ADS-B)

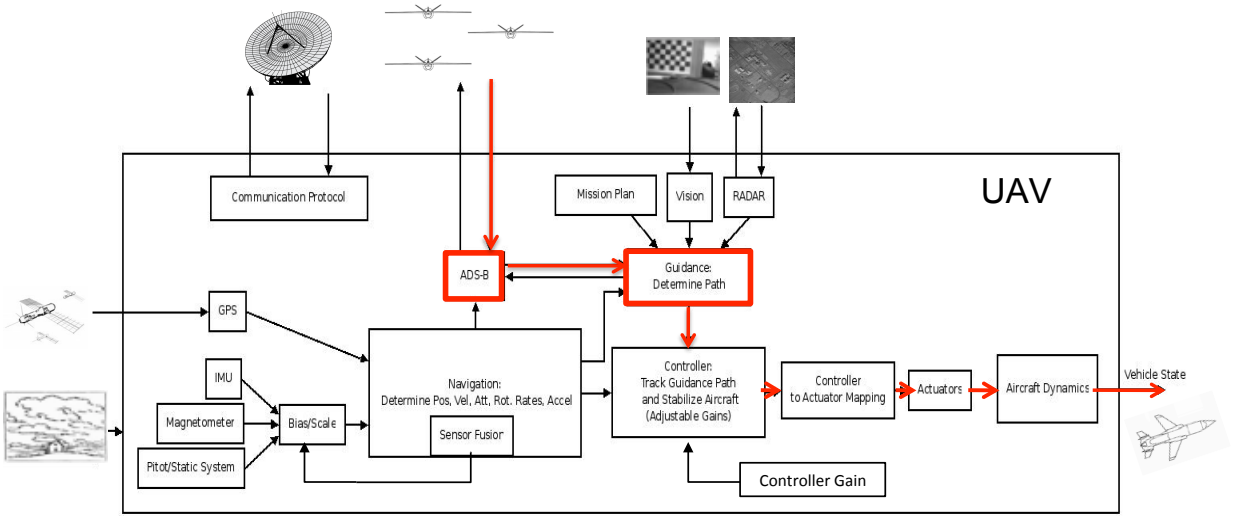


Figure 9: Flow of corruption due to ADS-B attack.

Figure 9 shows the components that can be affected by the ADS-B attack. This attack is an example of an application logic security vulnerability. ADS-B is an on-board component part of the next generation air traffic control system, which broadcasts information about an aircraft, such as position, heading, speed, and intent. For a UAV this system will mainly be used for environmental awareness and collision avoidance,

which is part of the navigation component of the autopilot. Since ADS-B is a broadcast system intended for all nearby aircraft, the data transmitted is not encrypted. This creates an easy attack point for false data injection. The ADS-B data is used for navigation by the UAV autopilot, and false ADS-B data can accordingly throw the UAV off track during a mission. Also, if the ADS-B data is unavailable while another aircraft is en route for collision, the survivability of the UAV is affected greatly.

Some of the possible attack methods are spoofing ADS-B data and jamming. A multilateration verification method can be used to detect simple false broadcasts of the ADS-B data. This method uses two ADS-B sensors to estimate the direction of the broadcast source by measuring the time difference between the reception of the broadcasts on the sensors. Even with this verification, spoofing is harder to detect if the attacker is knowledgeable about the validation techniques. Figure 10 shows an example of a way to beat the multilateration verification method. [11] $\hat{\Psi}$ and $\hat{\Phi}$ represent the estimated direction of the ADS-B signal and the Aircraft (from the ADS-B data) respectively. It is clear that an aircraft that has zero conflict with another aircraft can spoof an ADS-B signal in order to fake an expected collision course, which will cause the own aircraft to change its course.

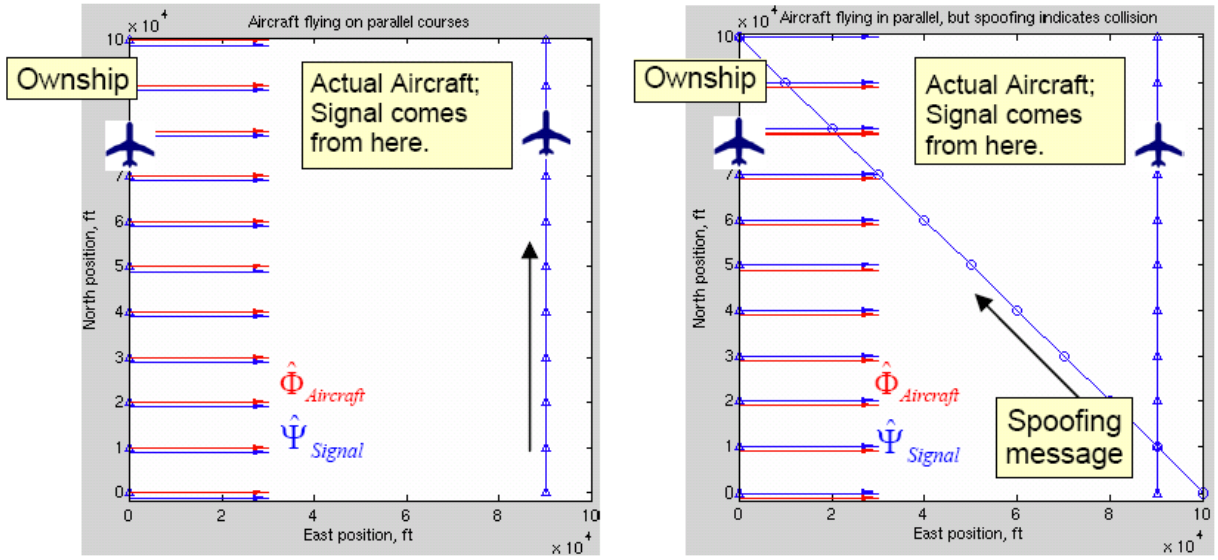


Figure 10: Example of failure of multilateration verification of ADS-B signal.

VI.B.5. Fuzzing

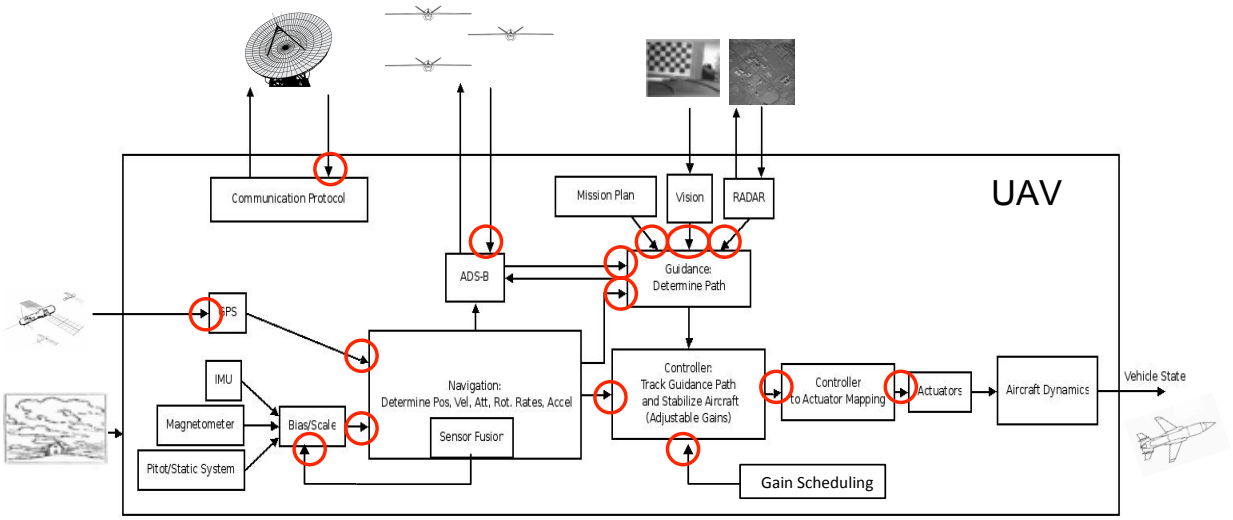


Figure 11: Attack points of fuzzing attack.

Figure 11 shows the attack point of the fuzzing attack. This attack is an example of a control system security and application logic security vulnerability. The concept of software fuzzing can be applied to GNC algorithms. In the UAV autopilot system, random inputs with expected distributions are not uncommon, and Gaussian noise inputs are routinely accounted for. However, unexpected, invalid, or completely random inputs can cause unknown behavior. If an attacker can somehow access any of the data flow between components and corrupt them with junk values, it will cause unexpected problems for the autopilot system. The consequences for this type of attack could include aircraft instability, process lock-up, and invalid outputs to the next process.

Some of the possible attack methods are buffer overflow attacks, sending malicious packets with invalid payload data to the UAV, and adding malicious hardware between components. It is also possible to use malicious fuzzing as a tool to discover vulnerabilities. Intentional white-box and black-box fuzzing tests could be performed on the system to determine the robustness of the GNC algorithms. [12]

VI.B.6. Digital Update Rate

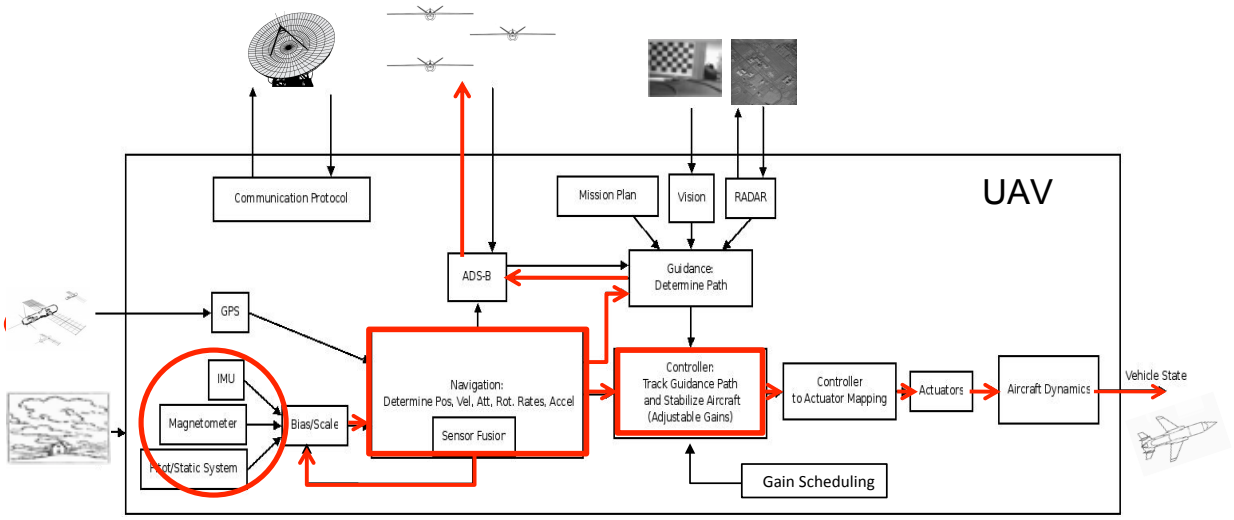


Figure 12: Flow of corruption due to digital update rate attack.

Figure 12 shows the components that can be affected by the digital update rate attack. This attack is an example of a control system security and application logic security vulnerability. UAV autopilots are digital computers and, accordingly, any inputs or outputs of the autopilot are discretized. This means that any continuous inputs to the autopilots are converted to digital inputs through discrete sampling. If the autopilot was designed with a continuous controller, that controller is also converted to a discretized form. For a discretized system, as the sample time increases the system becomes unstable/uncontrollable. [13] For data collection, longer sampling periods will increase the probability of data aliasing, as shown in Figure 13.

Some of the possible attack methods are changing the sampling time of analog-to-digital converters through buffer overflow or hardware manipulation and denial of service attacks that prevent the processor from running the controller or navigator at the desired update rate.

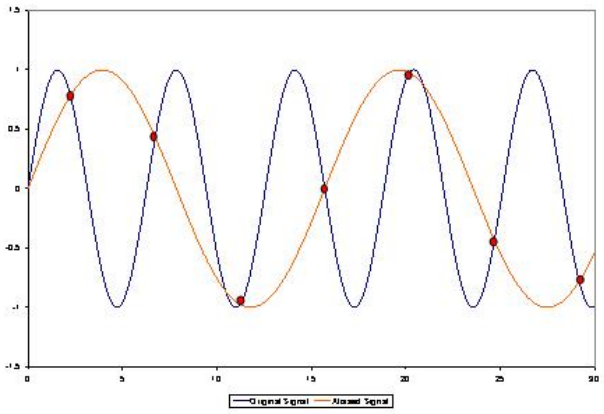


Figure 13: Example of aliasing of data due to insufficient sampling rate.

VII. Post-Attack Behavior Analysis

VII.A. Simulation Tools

VII.A.1. ScicosLab and Scicos

ScicosLab is a testbed that is very similar to MatLab and Simulink, but is an open source alternative. Like MatLab, ScicosLab allows easy matrix operations and the ability to execute mathematical functions and scripts. Scicos, like Simulink, is a block-based graphical simulation suite designed for control system design and simulation. We use Scicos to run ScicosLab, C, and C++ code underlying the graphical form. Since numerical analysis of aircraft dynamics requires large mathematical calculations, ScicosLab and Scicos are used regularly in our lab to carry out these simulations.

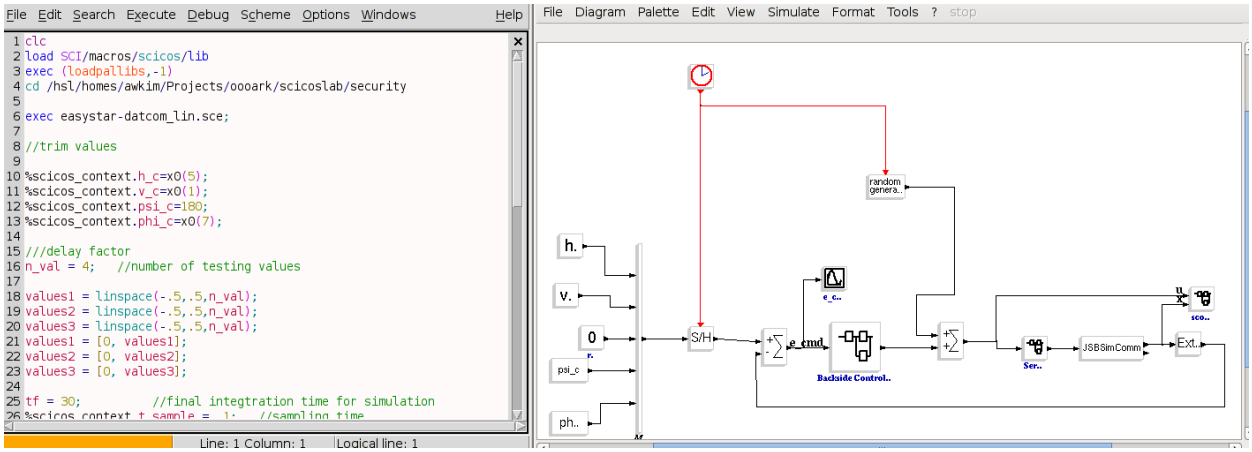


Figure 14: Screen shot of ScicosLab interface on the left and Scicos interface on the right.

VII.A.2. Arkscicos

Arkscicos (Autonomous Robotics Kit Toolbox for Scicos) is a software library for modeling and simulating unmanned flight which was developed by the Purdue Hybrid Systems Lab. Arkscicos was developed to be used with ScicosLab and Scicos. This software library features:

- Hardware In the Loop (HIL) interfaces
- Sensor models including Global Positioning System (GPS), Inertial Navigation System (INS), GPS/INS, Vision, and Magnetometer
- Visualizations
- Data logging
- High fidelity 6DOF aircraft dynamics

VII.A.3. JSBSim

JSBSim is a numerical analysis tool which simulates high-fidelity, 6-degree-of-freedom aircraft dynamics. The relevant equations of motion are derived by Stevens and Lewis [13]. JSBSim has a large library of aircraft models available, and can load custom models. JSBSim also has the ability to incorporate weather conditions into its numerical analysis.

Our lab has a small, easy to fly R/C airplane called the Easy Star that has been adapted for use as a UAV. The Easy Star uses the rudder and elevator as control surfaces. The rudder, elevator, and throttle comprise the three inputs into the system. Our custom autopilot is used on the Easy Star, so we have developed an accurate JSBSim model of it to use in developing and testing the autopilot before live flight. The Easy Star is shown in Figure 15 being tested in the wind tunnel at Purdue. We will use the Easy Star model to test our identified cyber threats and vulnerabilities of the current UAV autopilot system.

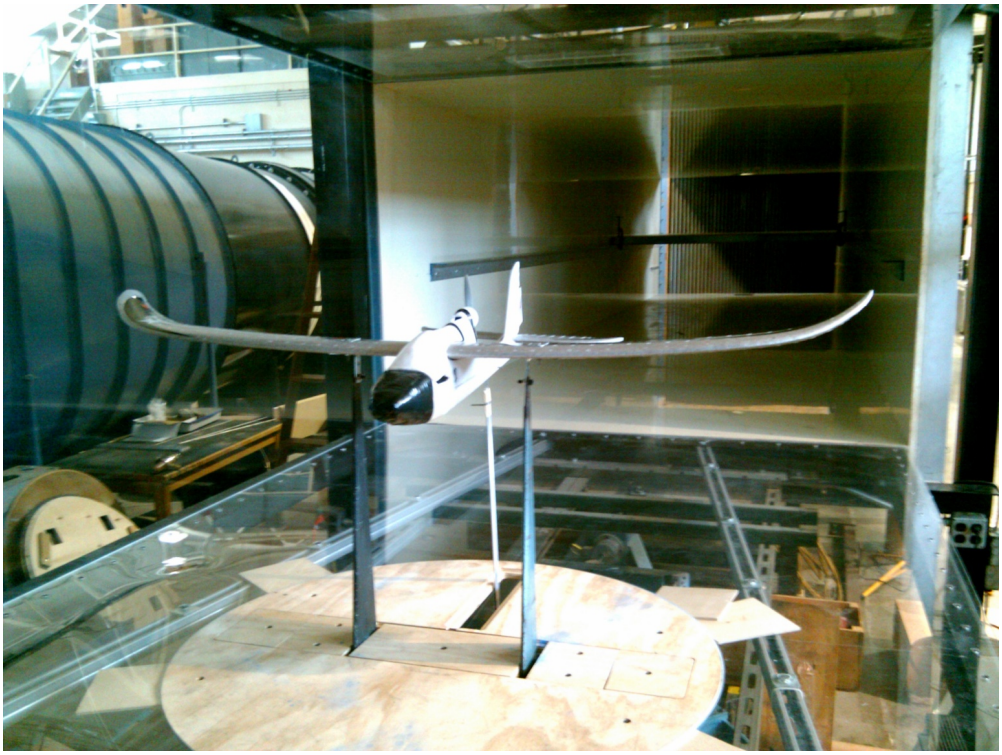


Figure 15: Picture of the Easy Star being tested in the Boeing wind tunnel at Purdue.

VII.A.4. Flight Gear

Flight Gear is an open source flight simulator which we use to visualize numerical simulations. Flight Gear can receive data from JSBSim and display the physical actions of an aircraft accurately in a simulated real world environment. Figure 16 demonstrates a simulation of a plane flying through San Francisco.

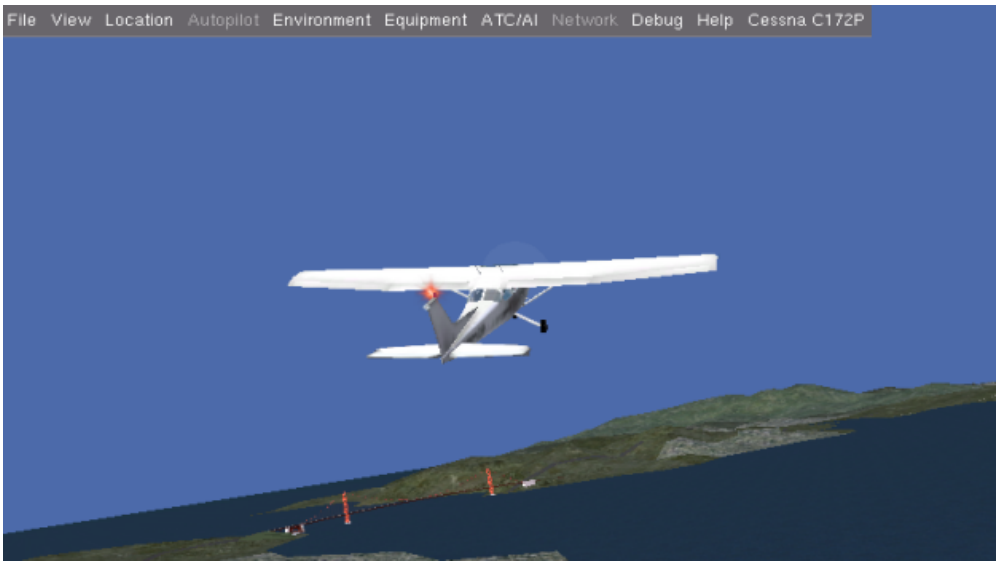


Figure 16: Screen shot of a flight simulation on Flight Gear.

VII.A.5. Purdue HSL Analysis Tool

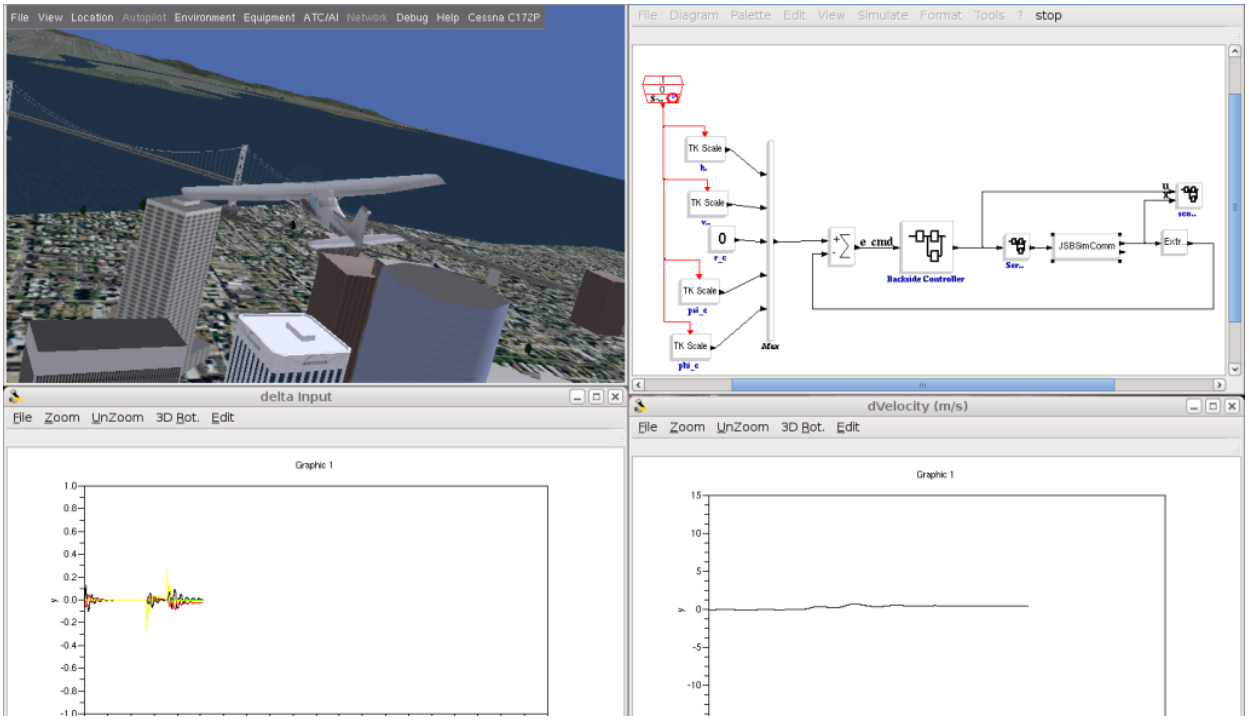


Figure 17: Screen shot of HSL Analysis Tool being used.

At the Purdue Hybrid Systems Lab (HSL), we have developed a high-fidelity simulation testbed for fully-autonomous unmanned aircraft using all of the aforementioned tools. We can study and test many aspects of the unmanned aircraft, such as command and control, sensors, and stability. Figure 17 shows a simulation being run using the Purdue HSL Analysis Tool, which can log any data while visually simulating the aircraft motion. We have also helped to develop a ground station for fully autonomous UAVs which can be seen in Figure 18. Again, the motion of the UAV is visually simulated while all the data of the simulation can be logged. The ground control provides a display of UAV information such as position, speed, heading, and sensor data, along with the control over the UAV. With our Purdue HSL Analysis Tool, we performed numerical analysis on the effects of the identified cyber attacks on the UAV autopilot system.

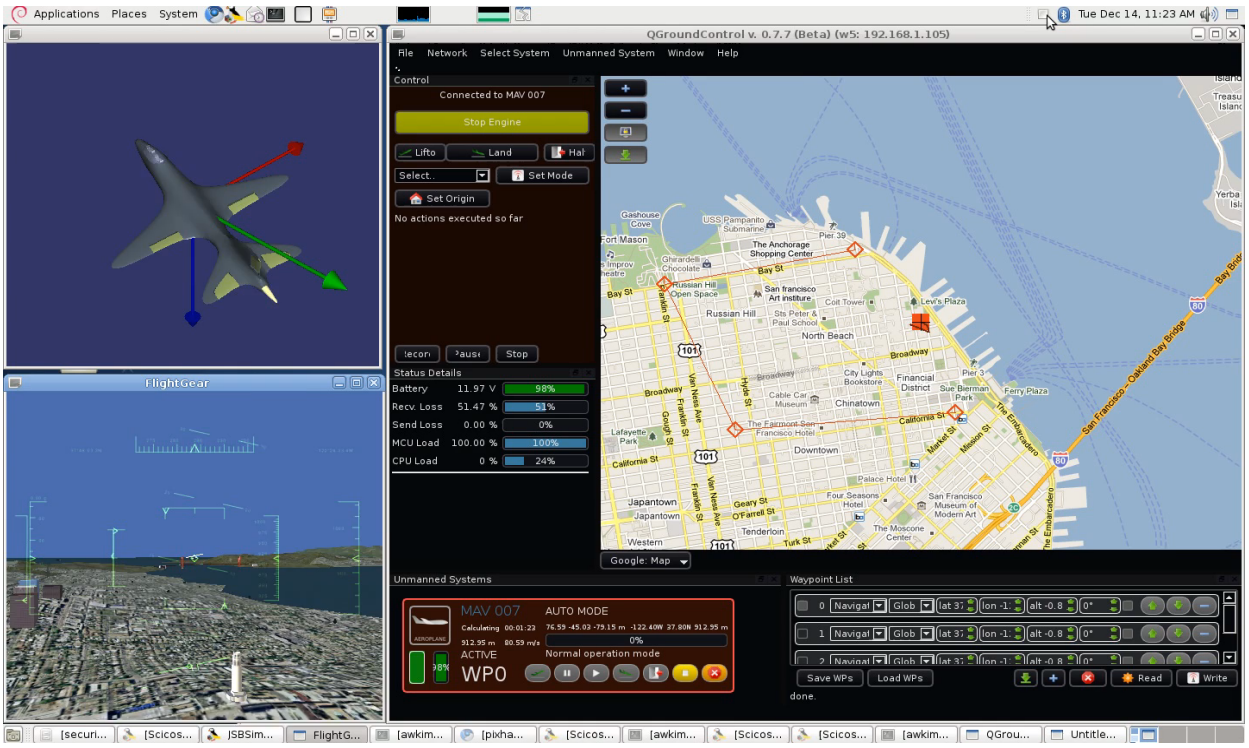


Figure 18: Screen shot of the ground control software (QGroundControl) and flight simulation.

VII.B. Gain Scheduling

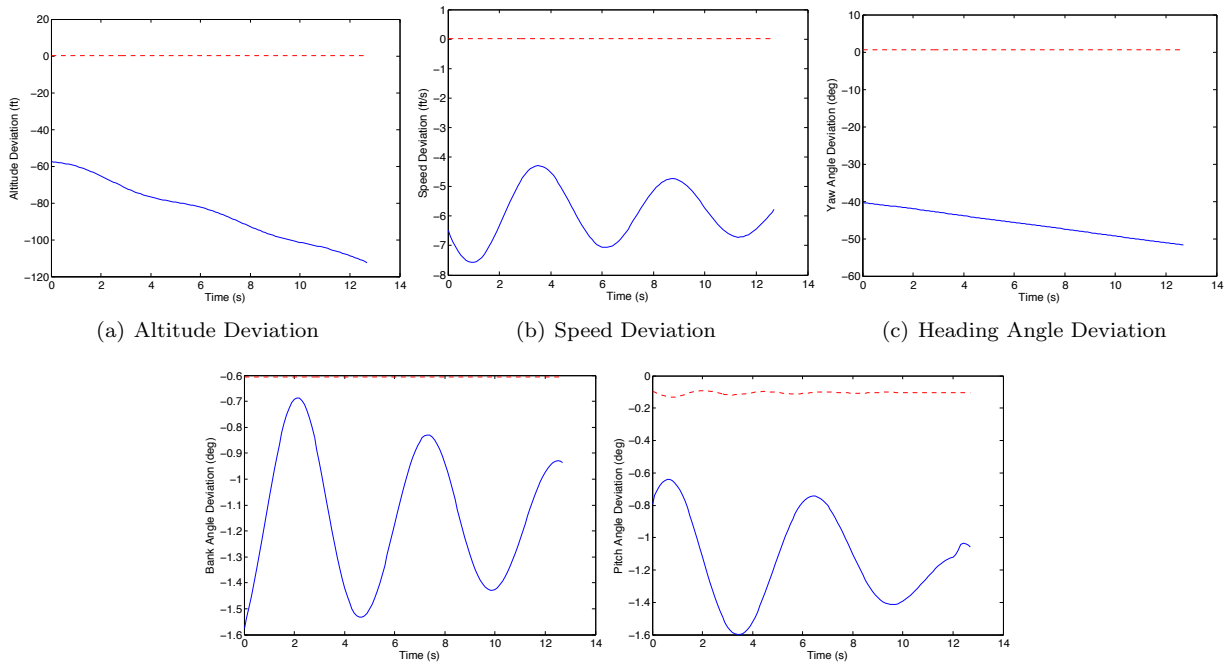
VII.B.1. Analysis

The gain scheduling attack can be used in several different ways as explained previously. For this study we have chosen to simulate the case where the UAV being attacked has several pre-programmed trim state stabilization gains for different flight conditions and the attacker causes a switch to a set of gains that does not match the current flight conditions. The autopilot is used mostly during the long period cruise portion of a mission, which gives the attacker a big window for an opportunity to use the gain scheduling attack.

The simulation was run on a flight of the Easy Star UAV (Figure 15) model in a trim cruise condition of:

- Altitude: 1000 ft
- Speed: 45 ft/s
- Heading: 180 deg
- Bank Angle: 0 deg

Five values were measured during the UAV cruise: the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle. Measuring these deviations shows the stability of the UAV under the autopilot control. The gain scheduling attack was simulated by flying the Easy Star at a flight condition far removed from trim. This was done to confirm the danger of operating a UAV with the wrong set of gains. The measured deviations and the visualization of the flight clearly show that this attack is detrimental to the stability of the flight compared to the normal case, confirming our hypothesis. A typical response of the UAV is plotted in Figure 19.



(a) Altitude Deviation (b) Speed Deviation (c) Heading Angle Deviation

(d) Bank Angle Deviation (e) Pitch Angle Deviation

Figure 19: Sample data for the gain scheduling attack. The red dotted line represents the normal case and the blue solid line represents the case of an attack. This sample data was collected after the start of the simulation. As such, zero time here represents the start of the data collection and not the start of the simulation.

VII.B.2. Sensitivity Study

In order to identify the most effective use of the gain scheduling attack, a sensitivity study was performed by simulating the flight of Easy Star at varying flight conditions and keeping the constant control gains. Each of the flight conditions (altitude, speed, bank angle) were varied to generate 125 data points. The following shows the values used:

- Flight Altitude: 5 evenly spaced values between -50% and 50% offset from normal trim, and normal case
- Flight Speed: 5 evenly spaced values between -50% and 50% offset from normal trim, and normal case
- Bank Angle: 5 evenly spaced values between -$\pi/6$ and $\pi/6$ offset from trim, and normal case

Five values were were measured during the UAV flight which are the squares of the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle. These values were plotted to identify which flight stability factor is most sensitive to which flight condition change (gain set change). This was done by creating plots in the following way:

1. Pick a design variable (in our case, flight altitude, flight speed, or bank angle).
2. Create a plot where the x-axis will represent the chosen design variable.
3. Choose a measurement variable (in our case, squares of the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle), which will be plotted on the y-axis against the chosen design variable.
4. Plot all the data that corresponds to each of the design variable values.
5. Calculate the means of the data set that corresponds to each of the design variable values and plot them.

6. Observe the trend of the mean values of the measurement data due to the design variable. If a linear trend line is used, the steeper slope of the trend line indicates higher sensitivity of the measurement variable to the design variable.

Figures 20, 21, 22, 23, and 24 show the sensitivity plot for altitude, speed, heading, bank angle, and the pitch angle respectively. By observing each of the plots, we concluded that for each stability factor, the following gain change attacks are most effective:

- Altitude Stability: Gain change for lower **altitude**
- Speed Stability: Gain change for higher **bank angle**
- Yaw Stability: Gain change for different operating **speed**
- Roll Stability: Not one attack point is more vulnerable
- Pitch Stability: Gain change for lower **bank angle**

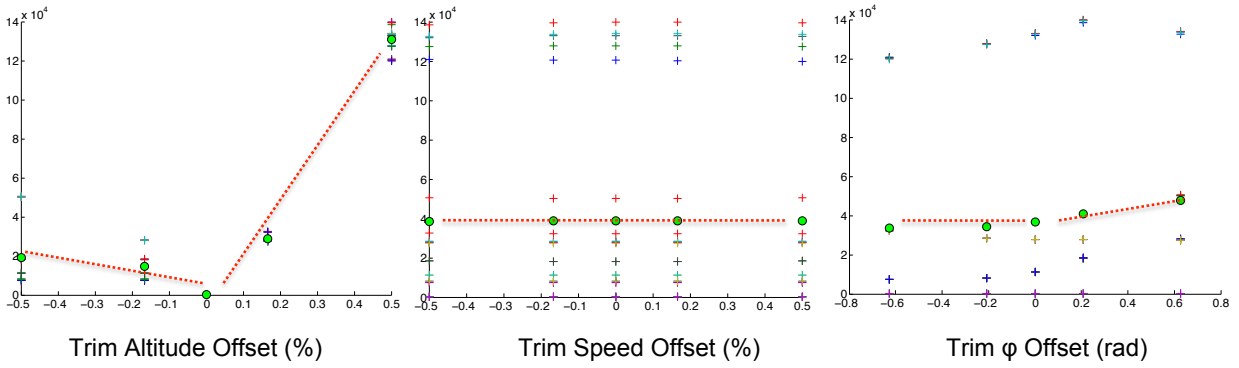


Figure 20: Sensitivity comparison plots for gain scheduling attack effect on altitude stability. The y-axis represents the square of deviation of altitude from trim condition (ft^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

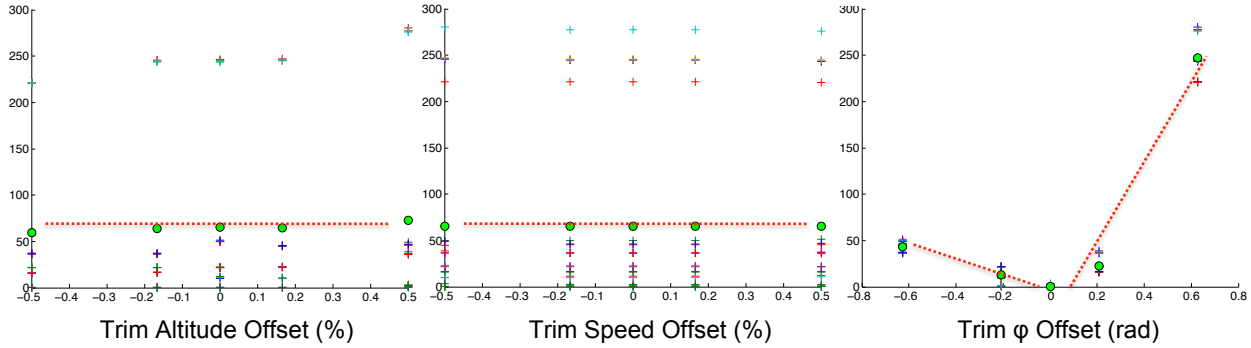


Figure 21: Sensitivity comparison plots for gain scheduling attack effect on speed stability. The y-axis represents the square of deviation of speed from trim condition (ft^2/s^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

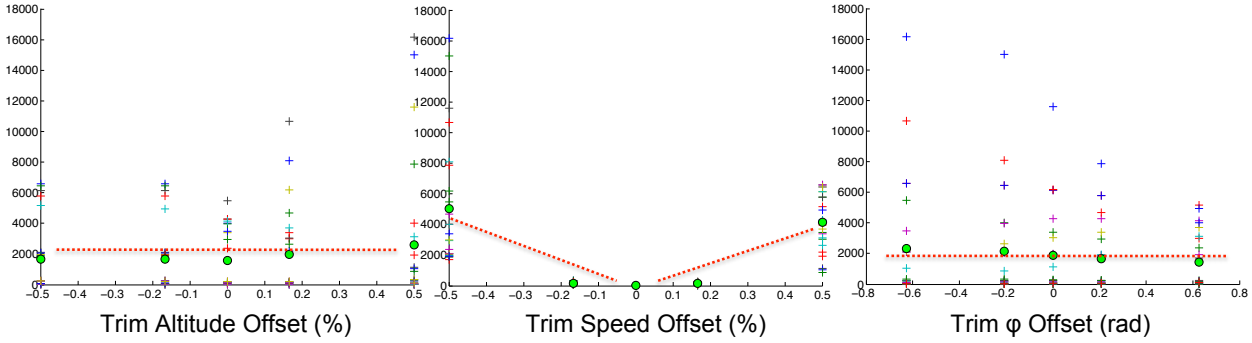


Figure 22: Sensitivity comparison plots for gain scheduling attack effect on heading stability. The y-axis represents the square of deviation of heading from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

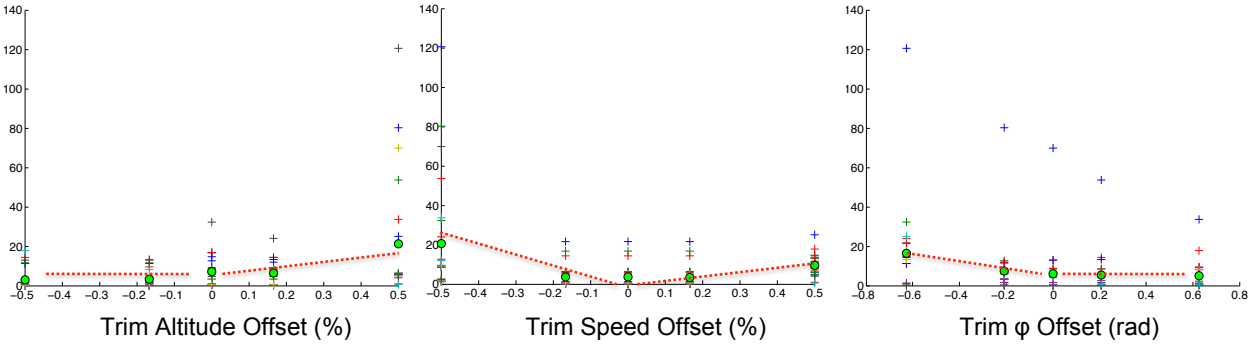


Figure 23: Sensitivity comparison plots for gain scheduling attack effect on bank angle stability. The y-axis represents the square of deviation of bank angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

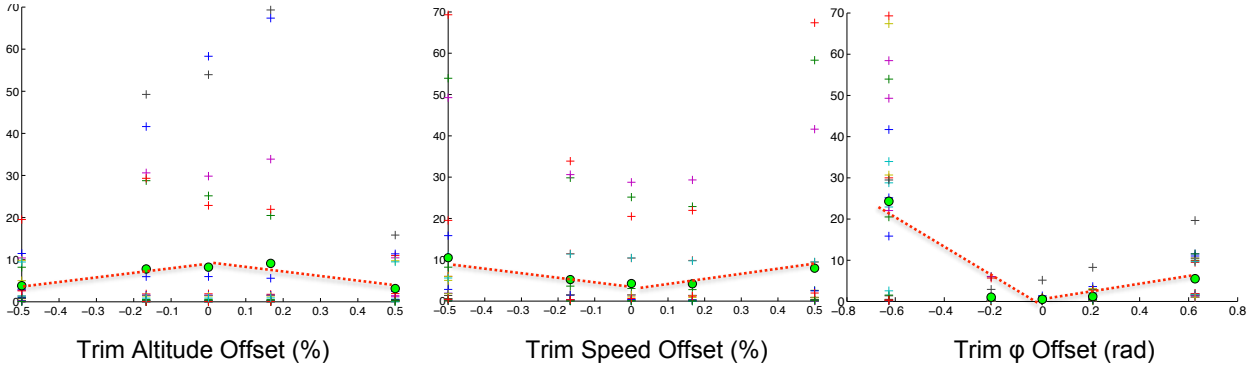


Figure 24: Sensitivity comparison plots for gain scheduling attack effect on pitch angle stability. The y-axis represents the square of deviation of pitch angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

VII.C. Actuator/Sensor

We have carried out the numerical analysis of the fuzzing attack on the actuators which is precisely the actuator/sensor attack. See the fuzzing section for the results.

VII.D. GPS

The study into the GPS vulnerabilities of autopilot systems proved to be a large undertaking in and of itself. We are currently carrying out research into this subject matter which is out of the scope of this paper. We do report that the GPS spoofing can lead a UAV astray and with proper techniques, the attack is undetectable while giving limited control over the destination of the UAV to the attacker.

VII.E. Fuzzing

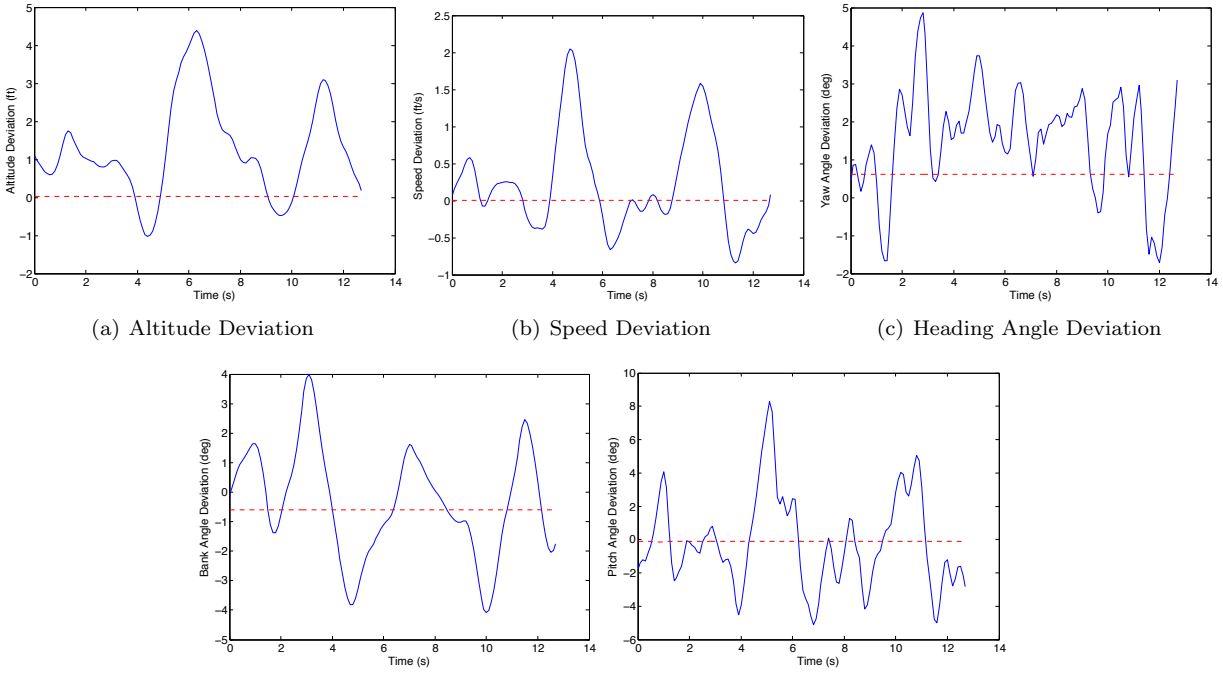
VII.E.1. Analysis

The fuzzing attack can be injected at several different points in the autopilot data flow. For this study we have chosen to simulate the case where the attacker injects random inputs to the actuators (i.e. corrupt the data from the controller to the actuators). This was done to also simulate the actuator attack.

The simulation was run on a flight of the Easy Star UAV (Figure 15) model in a trim cruise condition of:

- Altitude: 1000 ft
- Speed: 45 ft/s
- Heading: 180 deg
- Bank Angle: 0 deg

Five values were measured during the UAV cruise which are the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle. Measuring these deviations shows the stability of the UAV under the autopilot control. The fuzzing attack was simulated by injecting random Gaussian noise into each of the actuator inputs (throttle, elevator, rudder) at a 10 Hz rate. This was done to confirm the danger of operating a UAV under attack. The measured deviations and the visualization of the flight clearly show that this attack affects the stability of the flight compared to the normal case, confirming our hypothesis. A typical response of the UAV is plotted in Figure 25.



(a) Altitude Deviation (b) Speed Deviation (c) Heading Angle Deviation

(d) Bank Angle Deviation (e) Pitch Angle Deviation

Figure 25: Sample data for the fuzzing attack. The red dotted line represents the normal case and the blue solid line represents the case of an attack. This sample data was collected after the start of the simulation. As such, zero time here represents the start of the data collection and not the start of the simulation.

VII.E.2. Sensitivity Study

In order to identify the most effective noise injection point, a sensitivity study was performed by simulating the flight of Easy Star at the trim condition and injecting Gaussian noise with varying standard deviations to each of the actuators. The following shows the values used to create 125 data points:

- Throttle: 5 evenly spaced values from 0 to 0.2
- Elevator: 5 evenly spaced values from 0 to 0.2
- Rudder: 5 evenly spaced values from 0 to 0.2

Since this simulation was stochastic, Monte Carlo simulations of 10 runs were run for each of the test cases. Five values were measured during the UAV flight which are the average of the squares of the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle from the Monte Carlo runs. These values were plotted in the same manner as in the sensitivity study of the gain scheduling attack to identify which flight stability is most sensitive to which data corruption.

Figures 26, 27, 28, 29, and 30 show the sensitivity plot for altitude, speed, heading, bank angle, and the pitch angle respectively. By observing each of the plots we concluded that for each of the stability factors, the following injection points are most effective:

- Altitude Stability: **Rudder** input noise
- Speed Stability: **Rudder** input noise
- Yaw Stability: **Elevator** input noise
- Roll Stability: **Elevator** input noise
- Pitch Stability: **Rudder** input noise

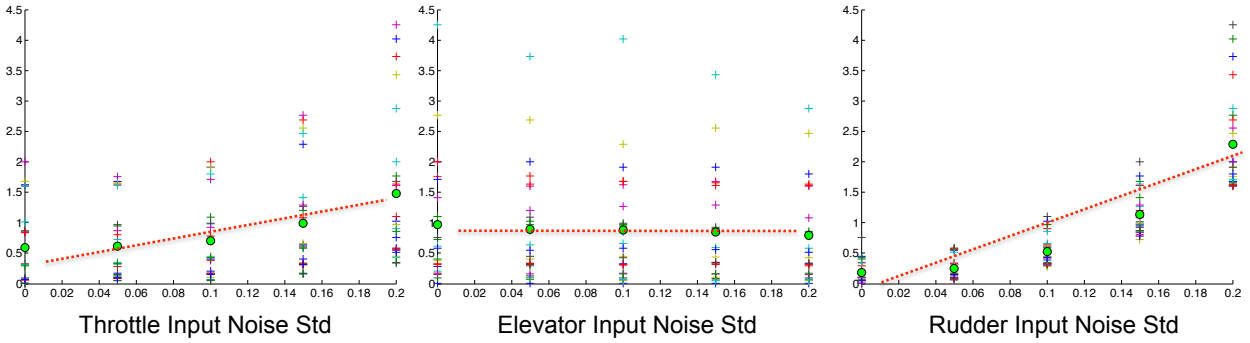


Figure 26: Sensitivity comparison plots for fuzzing attack effect on altitude stability. The y-axis represents the square of deviation of altitude from trim condition (ft^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

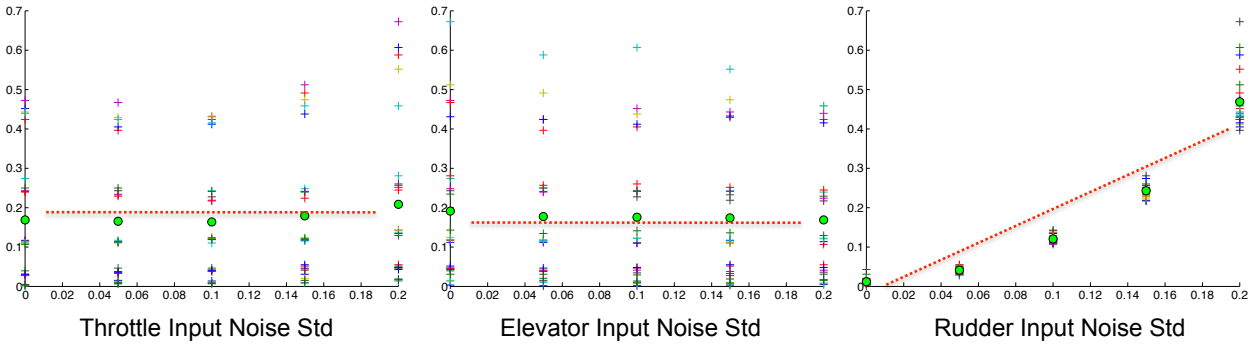


Figure 27: Sensitivity comparison plots for fuzzing attack effect on speed stability. The y-axis represents the square of deviation of speed from trim condition (ft^2/s^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

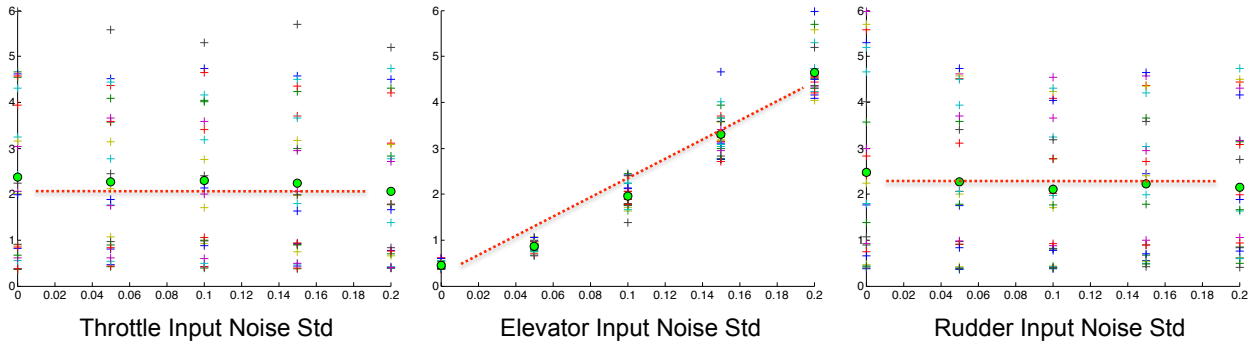


Figure 28: Sensitivity comparison plots for fuzzing attack effect on heading stability. The y-axis represents the square of deviation of heading from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

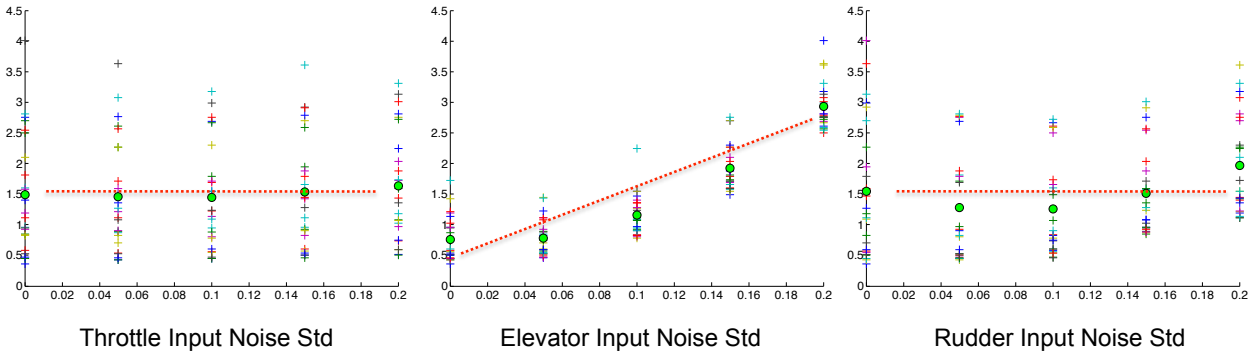


Figure 29: Sensitivity comparison plots for fuzzing effect on bank angle stability. The y-axis represents the square of deviation of bank angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

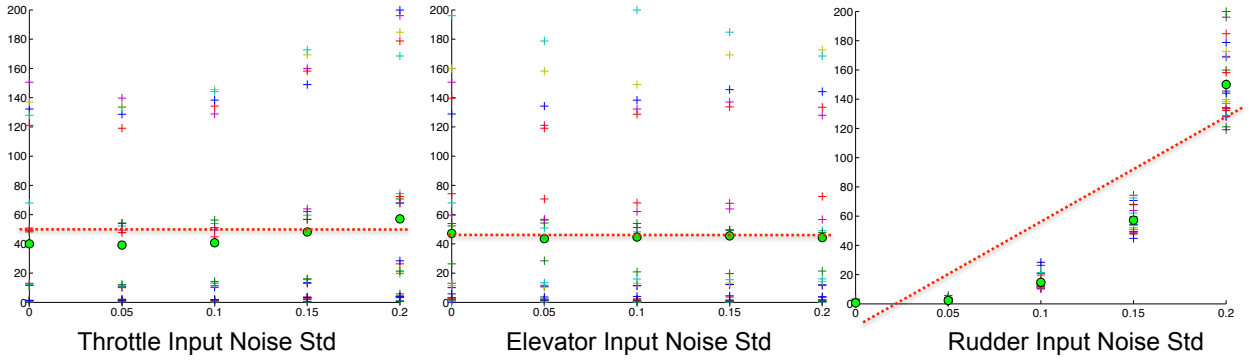


Figure 30: Sensitivity comparison plots for fuzzing effect on pitch angle stability. The y-axis represents the square of deviation of pitch angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

VII.F. Digital Update Rate

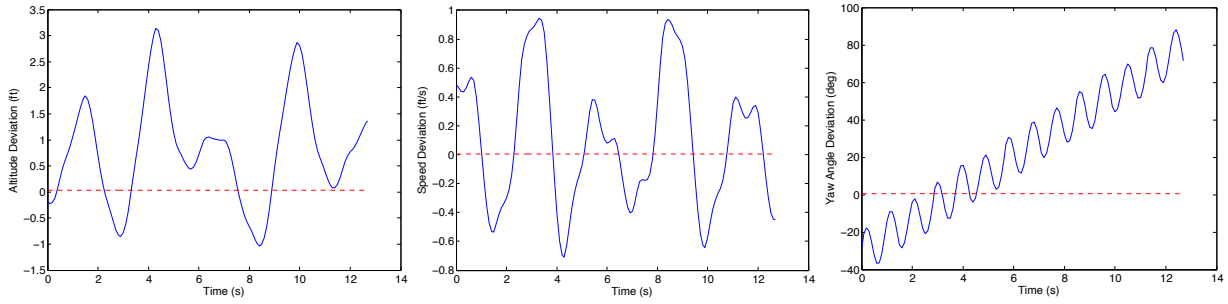
VII.F.1. Analysis

The digital update rate attack can be injected at several different points in the autopilot data flow. For this study we have chosen to simulate the case where the attacker delays inputs to the actuators (i.e. delay the data from the controller to the actuators).

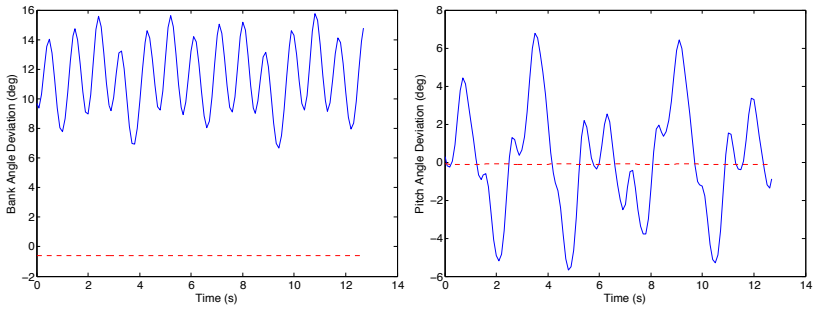
The simulation was run on a flight of the Easy Star UAV (Figure 15) model in a trim cruise condition of:

- Altitude: 1000 ft
- Speed: 45 ft/s
- Heading: 180 deg
- Bank Angle: 0 deg

Five values were were measured during the UAV cruise which are the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle. Measuring these deviations shows the stability of the UAV under the autopilot control. The digital update rate attack was simulated by delaying the inputs from the controller to each of the actuators (throttle, elevator, rudder). This was done to confirm the danger of operating a UAV under attack. The measured deviations and the visual of the flight clearly show that this attack affects the stability of the flight compared to the normal case, confirming our hypothesis. A typical response of the UAV is plotted in Figure 31.



(a) Altitude Deviation (b) Speed Deviation (c) Heading Angle Deviation



(d) Bank Angle Deviation (e) Pitch Angle Deviation

Figure 31: Sample data for the digital time update attack. The red dotted line represents the normal case and the blue solid line represents the case of an attack. This sample data was collected after the start of the simulation. As such, zero time here represents the start of the data collection and not the start of the simulation.

VII.F.2. Sensitivity Study

In order to identify the most effective data delay injection point, a sensitivity study was performed by simulating the flight of Easy Star at the trim condition and delaying the inputs to each of the actuators by varying times. The following shows the values used to create 125 data points:

- Throttle: 5 evenly spaced values from 0 to 0.2 seconds of delay
- Elevator: 5 evenly spaced values from 0 to 0.2 seconds of delay
- Rudder: 5 evenly spaced values from 0 to 0.2 seconds of delay

Five values were were measured during the UAV flight which are the squares of the deviations from the trim values of the altitude, speed, heading, bank angle, and the pitch angle. These values were plotted in the same manner as in the sensitivity study of the gain scheduling attack to identify which flight stability is most sensitive to which input delay.

Figures 32, 33, 34, 35, and 36 show the sensitivity plot for altitude, speed, heading, bank angle, and the pitch angle respectively. By observing each of the plots we concluded that for each of the stability factors, the following delays are most effective:

- Altitude Stability: **Rudder** input delay
- Speed Stability: **Rudder** input delay
- Yaw Stability: **Rudder** input noise
- Roll Stability: **Elevator** input delay
- Pitch Stability: **Rudder** input noise

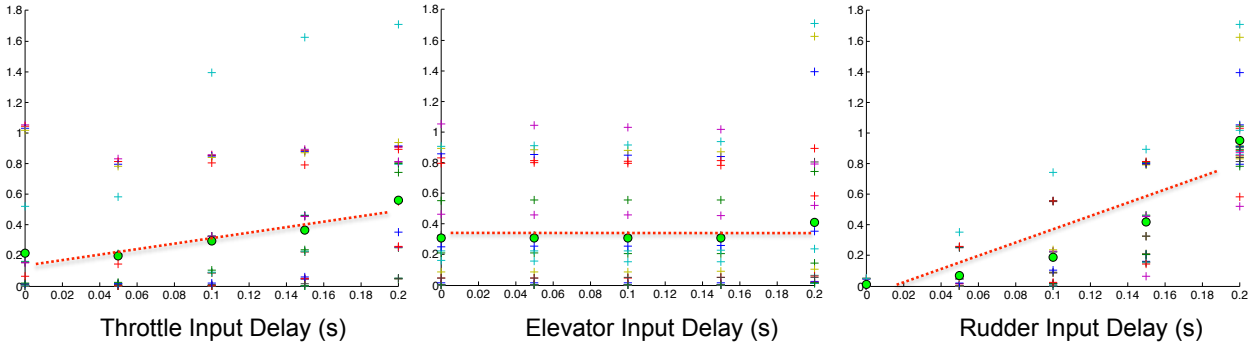


Figure 32: Sensitivity comparison plots for digital update rate attack effect on altitude stability. The y-axis represents the square of deviation of altitude from trim condition (ft^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

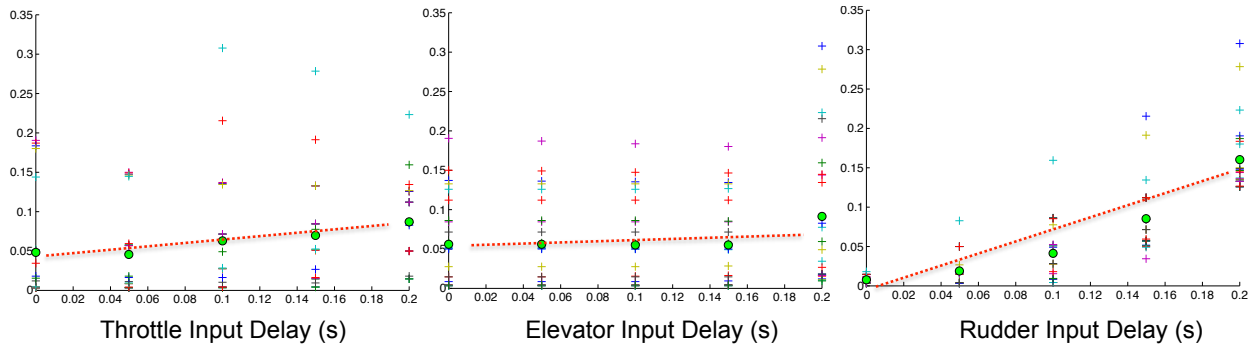


Figure 33: Sensitivity comparison plots for digital update rate attack effect on speed stability. The y-axis represents the square of deviation of speed from trim condition (ft^2/s^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

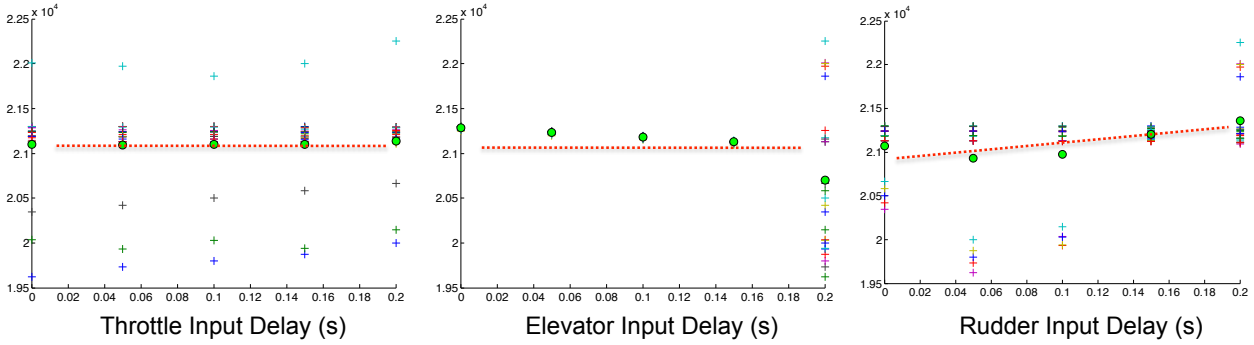


Figure 34: Sensitivity comparison plots for digital update rate attack effect on heading stability. The y-axis represents the square of deviation of heading from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

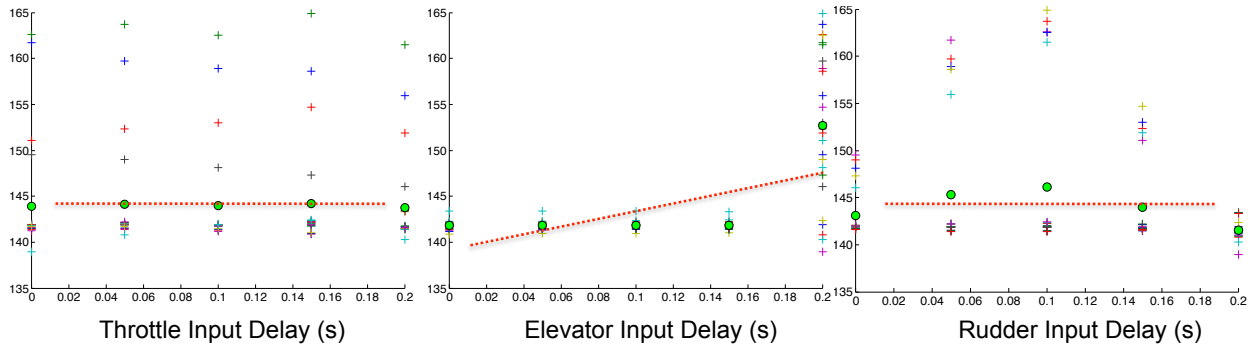


Figure 35: Sensitivity comparison plots for digital update rate attack effect on bank angle stability. The y-axis represents the square of deviation of bank angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

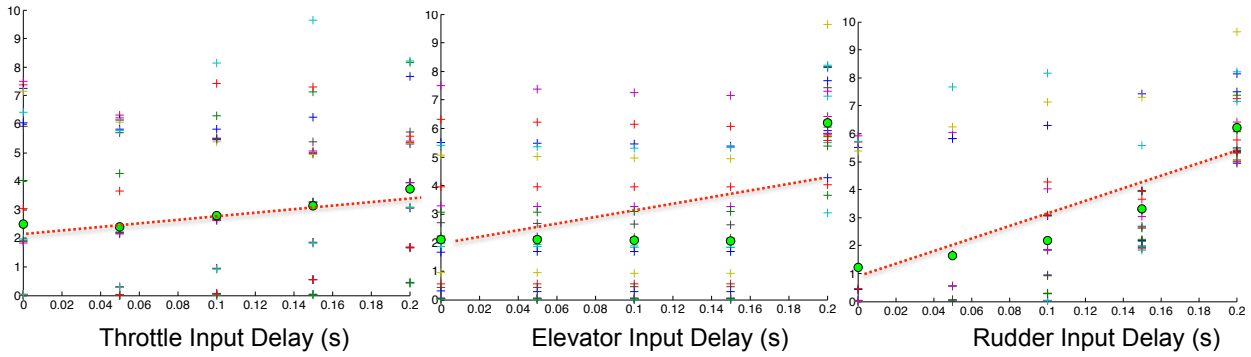


Figure 36: Sensitivity comparison plots for digital update rate attack effect on pitch angle stability. The y-axis represents the square of deviation of pitch angle from trim condition (deg^2). The green circle represents the mean of the + marks, which represent the result of each execution. The dotted line represents the approximate linear fit line of the mean values.

VIII. Cyber-Secure Architecture of Autopilot System

We propose to develop a system architecture that is more robust to these cyber attacks by augmenting the UAV autopilot system with a cyber-security monitoring component, called a *Supervisor*. This proposed architecture is shown in Figure 37. The *Supervisor*'s role is to detect and isolate abnormal or malicious activity within the autopilot system. It can report over the communication link but will remain inaccessible from the outside to help ensure it maintains security. It will read and monitor the other systems without being directly in the operation loop. This will protect it from attacks and problems that may propagate through the system. It will also have the ability to recognize compromised areas and reconfigure the autopilot system based on this knowledge. [14]

If malicious code were loaded that could affect operation of the autopilot system, this would trigger the supervisor since the behavior of the vehicle would not be as expected. For example, if a piece of code is loaded that flips the direction of the aileron feedback, the resulting behavior of the vehicle would not match that of the dynamic model, and would be caught by the supervisor. The supervisor may disable this aileron and fly with just one aileron. The performance may be reduced but the vehicle is able to safely operate.

If false information were supplied, such as a false ADS-B signal, the supervisor would recognize that the signal strength does not match the signal strength model for that broadcast range. Then, using probabilistic methods, the supervisor can estimate the likelihood that the signal is incorrect and report it to the ground station or other members of the network.

Possibly the hardest attack to identify is the direct control of the vehicle through commands, changes in

flight plan, or mission objective. While additional layers of encryption and authentication should be added to hinder this type of attack, it is still necessary to use the supervisor to aid in security. One idea is to monitor if these commands place the vehicle outside of an operational envelope or mission envelope based on the system model and mission model. [15] The operational envelope is the combination of limits within which the system can safely operate such as airspeed, g-force, altitude, etc. The mission envelope is the allowable deviation from the mission plan that still allows the objective to be effectively completed. If the command actually changes the mission envelope, then this could perhaps trigger a verification request by the supervisor via another secure communication method, and at the very least notify that a change has been issued.

So far we have proposed the detection of cyber attacks. We further propose to handle the reconfiguration and control of a compromised system. This control decision would be based on the mode that the system is placed into as a result of the attack. The outcome of the attack should also be considered as well. When deciding a course of action, possible outcomes could include but are not limited to:

1. Objective is still achievable
2. Objective is achievable with reduced performance
3. Vehicle is placed outside of its safe operating envelope
4. Vehicle is placed outside of the mission envelope
5. System is captured or destroyed
6. Objective is maliciously altered

An attack that still allows the objective to be reached may require no reconfiguration at all, while other outcomes may require the system to return home or in extreme cases safely remove itself from operation.

There are many ways to prevent and handle cyber attacks on unmanned system. We propose that adding the Supervisor as an additional layer of security at the system design, architecture, and estimation and control levels will help contribute to a more robust multiple-layer security design. [16]

For completeness, Figure 37 also shows a detailed diagram of UAV autopilot processes and interactions of a micro-controller based system. This diagram identifies potential areas that could be vulnerable to cyber attacks in an architecture that includes a supervisor. It is color coded by threat level. The supervisor, guidance, navigation, and control processes all exist in read only memory. This memory is only writable when the processor is reset. A write to read only memory (ROM) can typically be triggered by a button on-board, a power cycle, or an external communication pin. Using the external communication pin would allow remotely programming the system but this is strongly discouraged as it would also allow a cyber attacker to reprogram the system. Assuming that the internal processes are not compromised, we can rely on the supervisor process to monitor reading to and writing from the Random Access Memory (RAM). This is where system critical information such as the vehicle state, flight plan, and communication buffers are stored.

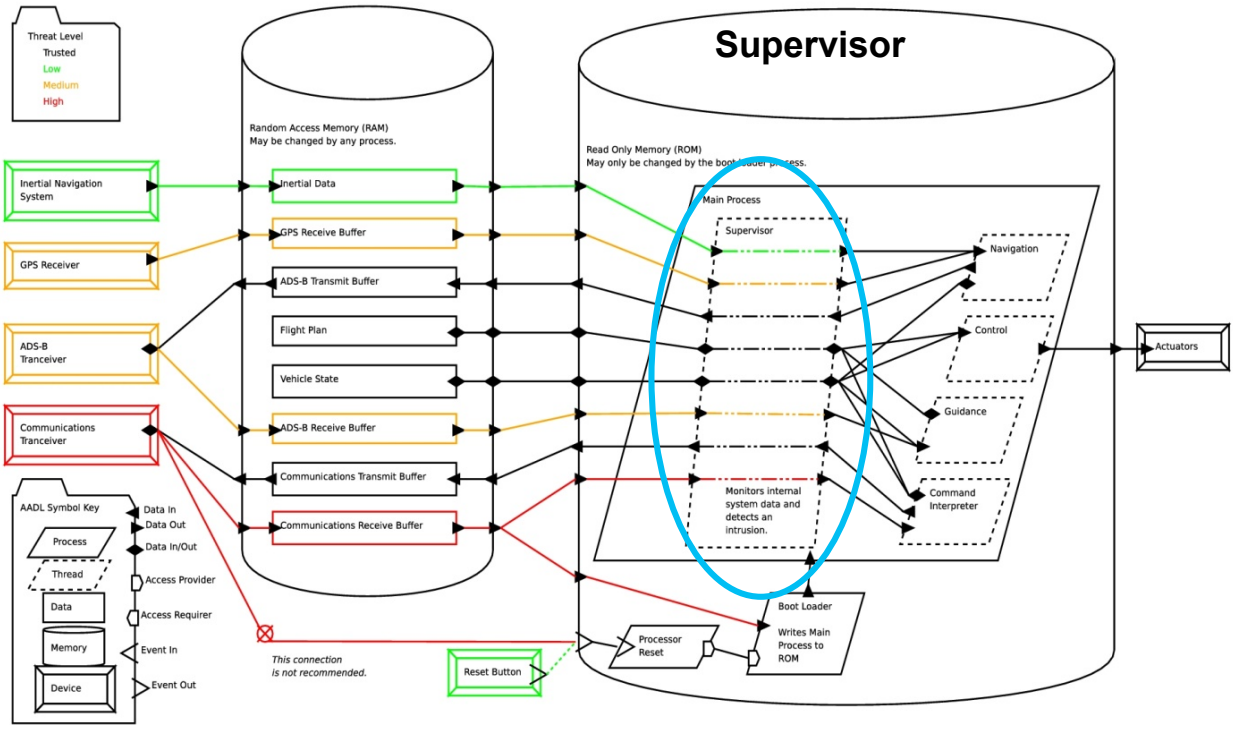


Figure 37: Data flow diagram of the UAV autopilot system with the *Supervisor*

Conclusion

Our final goal is to develop the *Supervisor* to be introduced into the current UAV autopilot system, which will make the system robust to cyber attacks. In order to do so we have several tasks planned which include:

- **Carry out GPS/INS Analysis**: So far we have carried out numerical analysis on GPS attack scenarios of a simple case without coupling of aircraft dynamics. We will develop a more sophisticated and accurate model to simulate the GPS attack being applied to a realistic scenarios by introducing actual aircraft dynamics. A sensitivity study for this type of attacks will also be carried out.
- **Carry out ADS-B Analysis**: In order to perform a numerical analysis on an ADS-B attack scenarios, we will first develop a collision avoidance algorithm. Then, a numerical analysis will be carried out simulating multiple aircraft. A sensitivity study for this type of attacks will also be carried out.
- **Study more sophisticated attack scenarios**: So far we have identified and analyzed simple cyber attacks that involve a single point of attack or a single method. We will study more sophisticated attacks such as the ones that utilizes multiple points of attack or multiple methods. We will further look into coordinated attack possibilities where the attacker uses several attacks in a certain manner to induce more effective faults into the autopilot system. We will also consider the disguised attack possibilities where the attacker can mask an attack to induce a false reaction from the autopilot in order to remedy the attack.
- **Analytical analysis**: We will also look analytically look for cyber attack method. For an example, certain Kalman filtering algorithms might be vulnerable to a special form of induced error in measurements which cannot be detected.
- **Develop metrics for cyber attacks**: So far a metric for measuring either a likelihood (or probability) or a damage potential of cyber attacks on a UAV autopilot does not exist. Developing these metrics is very important since it can be used to design the cyber-secure autopilot architecture.

- **Develop cyber attack detection algorithms**: One of the role of the *Supervisor* is to detect and isolate any cyber attacks. The Hybrid Systems fault detection approach can be used to develop a robust detection capability.

- **Develop the *Supervisor***: With all the necessary research completed, we can move onto actually developing the *Supervisor* which will play the main role in the cyber-secure UAV autopilot.

Acknowledgments

We would like to thank Sypris electronics for supporting this work. We would also like to thank the development communities of ArduPilotMega, PixHawk, and ScicosLab/SciLab.

References

[1]Frost and Sullivan, "Study Analysing The Current Activities in The Field of UAV," Tech. rep., European Commision Enterprise and Industry Directorate-General, 2007.

[2]Yochim, J. A., *The Vulnerabilities of Unmanned Aircraft System Common Data Links to Electronic Attack*, Master's thesis, U.S. Army Command and General Staff College, Fort Leavenworth, Kansas, Jan 2010.

[3]Army UAS CoE Staff, "U.S. Army Roadmap for UAS 2010-2035," Tech. rep., U.S. Army UAS Center of Excellence (ATZQ-CDI-C), 2010.

[4]Clapper, J. R., Young Jr., J. J., Cartwright, J. E., and Grimes, J. G., "Unmanned Systems Roadmap 2007-2032," Tech. rep., Memorandom for secretries of the military departments, Dec 2007.

[5]Donley, M. B. and Schwartz, N. A., "United States Air Force Unmanned Aircraft Systems Flight Plan 2009-2047," Tech. rep., United States Air Force, May 2009.

[6]Cowan, C., Wagle, F., Pu, C., Beattie, S., and Walpole, J., "Buffer overflows: attacks and defenses for the vulnerability of the decade," *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, Vol. 2, 2000, pp. 119–129.

[7]Shea, D. A., "Critical Infrastructure: Control Systems and the Terrorist Threat," Tech. rep., CRS Report for Congress, 2004.

[8]Alberts, D. S., Garska, J. J., and Stein, F. P., "Network Centric Warfare: Developing and Leveraging Information Superiority," Tech. rep., DoD C4ISR Cooperative Research Program, Feb 2000.

[9]Sakamoto, N. S., "UAV Development and History at Northrop Grumman Corporation Ryan Aeronautical Center," SI4000 SUMMER 2004 UAV Brief, 2004.

[10]Hwang, I., Kim, S., Kim, Y., and Seah, C., "A Survey of Fault Detection, Isolation, and Reconfiguration Methods," *IEEE Transactions on Control Systems Technology*, Vol. 18, May 2010, pp. 636–653.

[11]Krozel, J. and Andrisani, I., "Bindependent ADS-B verification and validation," *AIAA 5th Aviation*, 2005, pp. 1–11.

[12]Godefroid, P., Levin, M., and Molnar, D., "Automated Whitebox Fuzz Testing," *15th Annual Network and Distributed System Security Symposium (NDSS)*, Feb 2008.

[13]Stevens, B. and Lewis, F., *Aircraft Control and Simulation*, Wiley, New York, NY, 1992.

[14]Liu, W. and Hwang, I., "Robust Estimation Algorithm for A Class of Hybrid Systems with Unknown Continuous Fault Inputs," *American Control Conference (ACC)*, 2010.

[15]Hwang, I., Stipanovic, D. M., and Tomlin, C. J., "Polytopic approximations of reachable sets applied to linear dynamic games and to a class of nonlinear systems," *Advances in Control, Communication Networks, and Transportation Systems Systems and Control: Foundations and Applications*, 2005.

[16]Dufrene Jr., W., "Mobile military security with concentration on unmanned aerial vehicles," *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*, Vol. 2, October 2005, p. 8 pp.