Simulation-Based Logic Bomb Identification and Verification for Unmanned Aerial Vehicles

Jake Magness, Patrick Sweeney, Scott Graham, Nicholas Kovach

Chapter 1

SLIVER: SIMULATION-BASED LOGIC BOMB IDENTIFICATION/VERIFICATION FOR UNMANNED AERIAL VEHICLES

Jake Magness, Patrick Sweeney, Scott Graham, Nicholas Kovach

Abstract

This paper offers a novel approach to finding logic bombs hidden within UAV autopilot code without access to source code by executing mission runs in a software-in-the-loop simulator (SITL), and defining safe UAV operating areas in a larger region. The approach uses pre-planned flight paths as a baseline, greatly reducing the input space that must be searched to have confidence that the UAV will not encounter a trigger condition during its mission. This article discusses the process for creating a logic bomb in the ArduPilot autopilot software, test flight profiles, log file parsing, and analysis of the methodology. The logic bomb detection methodology created under this effort can accommodate multiple flight profiles and parses through the corresponding log files to create a safety corridor through which the UAV is able to safely traverse through with a 95% level of confidence. By utilizing this methodology or a similar approach, UAV operators and planners alike are afforded increased confidence that the aircraft will operate normally throughout the duration of a mission given proper test configuration.

Keywords: Logic Bomb, Trigger Condition, ArduPilot, Unmanned Aerial Vehicle (UAV), Software-in-the-loop (SITL)

1. Introduction

Military and civilian institutions are continually finding new use cases for UAVs. One of the many use cases that now exist for UAVs lies within the domain of critical infrastructure. UAVs offer cost effective and efficient solutions to a wide variety of challenges that exist within the various critical infrastructure sectors. For example, UAVs are now utilized for forest fire monitoring, power line inspection, and are even becoming part of the critical infrastructure as communications nodes [6]. With the increased usage and the foreseeable future reliance on UAVs as a key component of critical infrastructure systems, the security of these devices grows in importance.

There are currently a wide variety of exploits that have been developed and successfully employed against UAVs. These exploits are primarily traditional network-based attacks that target the Wi-Fi connectivity of the device to interrupt command and control functions, GPS spoofing or exploitation of the base station to name a few examples [15, 20, 10, 9, 17, 22]. These documented exploits pose a clear threat to each of the critical infrastructure sectors. Fortunately, many of these exploits are a result of poor practice and can be easily mitigated through the use of Wi-Fi passwords or secure protocols. There are more nefarious exploits that could potentially exist for UAVs however. As has been already demonstrated by exploitation of the United States Military and the United States Department of Homeland Security UAVs, some companies are willing to implant malicious code into the source code of their devices [17, 22]. These pieces of malicious software can fit multiple definitions but this paper is concerned primarily with malicious implanted logic, whether by the manufacturer or some other malicious actor along the way, that is triggered when a certain set of conditions is reached—a logic bomb. This leaves the question: how can a user know their device is free of logic bombs?

For sensitive applications such as critical infrastructure monitoring, there is a distinct need for UAVs supporting those missions to operate flawlessly. As previously discussed, corporations are willing to implant malicious logic into their UAV software. Depending on the intent of a malicious actor, that malicious logic could impact or even intentionally target critical infrastructure related UAV missions, affecting national security interests. For this reason, there is a need for a methodology that is designed to search for hidden malicious code trigger conditions.

The logic bomb detection methodology framework that has been developed is perfectly suited for tasks in which a UAV operates autonomously along a predefined flight profile or manually within a defined boundary

area. In these scenarios, this methodology assures to a very high degree of confidence that there are no malicious logic bomb trigger conditions present. Additionally, the developed methodology addresses a key issue in UAV logic bomb detection: restricted access to source code. Given that the majority of small UAVs are procured from outside commercial entities, it is of vital importance that a logic bomb detection methodology specifically tailored for UAVs has the ability to operate without source code access.

2. Background

This section provides a brief background on various topics relating to UAV usage in critical infrastructure applications, logic bombs, current logic bomb detection methodologies, logic bomb relation to UAV software, and blackbox testing. As UAVs continue to take on more important roles within the critical infrastructure domain, it is essential that these devices are secured from a wide variety of threats, including logic bombs.

2.1 UAV Critical Infrastructure Applications

The Department of Homeland Security has identified 16 different areas as critical infrastructure sectors [24]. Within each of these sectors, UAVs are becoming essential tools as they provide distinct advantages over previous critical infrastructure monitoring and management means. These advantages include the ability to fly into high-risk areas, fly with high fidelity navigational precision, lower operational cost, the ability to utilize different sensors depending on mission type, the ability to capture large-scale images, and the ability to create overlapped images [1]. These benefits have enabled various new use cases for UAVs in all 16 critical infrastructure sectors.

One such use case is emergency response in natural disaster scenarios. These scenarios can range from forest fires, to hurricane and tornado aftermath, or to any other post-natural disaster response scenario. [1]. Previously, operations such as this would be extremely dangerous or impossible to do on foot, or very expensive to do by other aerial means such as helicopter or airplane. UAVs have become an essential tool that can provide comparable or superior results at a reduced cost and risk.

UAVs have found new applications is the energy sector as well. They are used by oil and gas companies to not only survey proposed drilling sites but to then monitor the infrastructure for leaks or other issues including malicious attacks [6]. Timely and accurate monitoring of these sites is critical as the compromise of essential energy grid facilities is

a direct threat to natural security. UAVs provide a cost-effective and efficient way to monitor these facilities.

In short, the capabilities of UAVs have enabled them to fill a wide variety of roles within multiple critical infrastructure sectors, and their use is expected to increase. As the reliance on these devices grows, the interest of malicious actors grows as well. Developing a way to identify malware resident on the systems before it affects a mission is of critical importance.

2.2 Documented UAV Vulnerabilities

While UAV experimentation has been around for a considerable amount of time dating back to 1918, it hasn't been until recently that their increased affordability and capability have generated significant interest in the mainstream population [23]. As their use has become more pervasive, exploitation techniques have inevitably developed and these haven't yet been fully addressed. The attack vectors on currently available commercial UAVs vary widely and include attacks that target the communications protocol/network, GPS integrity, UAV operating system, and base station [15, 20, 22, 17]. Figure 1 shows the UAV/base station architecture and the associated vulnerabilities with each piece of with the architecture.

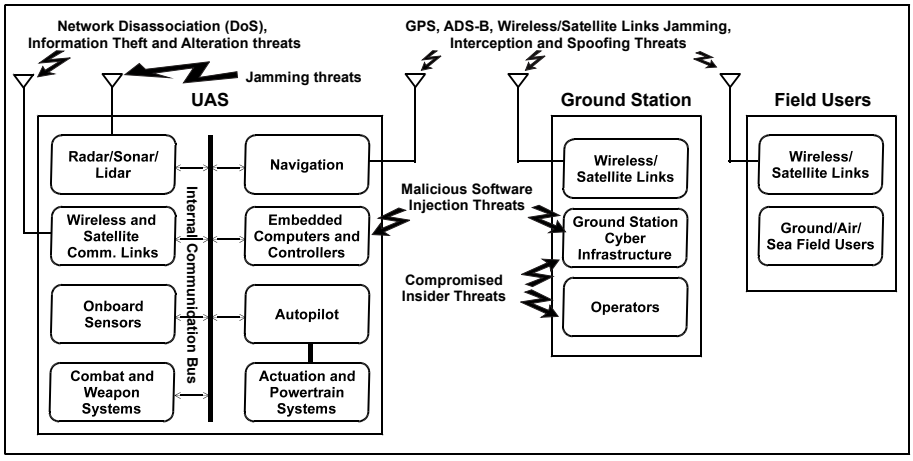


Figure 1: Attack Vectors: Potentially vulnerable systems that can be attacked on a UAV [15].

Figure 1 illustrates how each additional capability that has been added to UAVs predictably increases the attack surface and potential vulnerability. At this point, a large number of vulnerabilities stem from tradi-

tional network based attacks, protocol weakness, and operating system vulnerabilities. For example, researchers from the University of Brandenburg were able to take control of the Parrot AR.Drone 2.0 quadcopter by simply using the default FTP and Telnet root accounts [19]. Other vulnerabilities have been associated with commercial UAVs utilizing unsecured Wi-Fi networks as their command and control structure [10]. Furthermore, these researchers found that these Wi-Fi communication networks were vulnerable to nearly all standard Wi-Fi attacks such as ARP Cache poisoning, DHCP poisoning and MAC cloning [10]. In addition to inheriting network vulnerabilities, UAVs are susceptible to other types of attacks such as GPS spoofing. The focus of this research is a type of malware called a logic bomb that may be embedded within UAV autopilot code.

2.3 Logic Bombs

The term logic bomb is defined by [12] as "...malicious application logic that is executed, or triggered, only under certain (often narrow) circumstances." Despite the definition, early logic bombs were employed legitimately to render software unusable once a trial period had expired, requiring payment for a license before access is again granted [12]. However, it wasn't long before malicious uses of logic bombs came into the fold. A common scenario when a trigger condition was reached was for the logic bomb to delete files from a system or destroy network equipment [12, 25].

Logic bombs can be extremely difficult to detect, especially given the fact that they are often times planted by developers who have intimate knowledge of the system they are compromising [25]. Additionally, the focus of many malware analysis techniques is on the insertion of malicious code into running applications rather than the presence of malicious code existing in what would be considered "good" or "clean" code [2]. The insider threat dynamic of logic bomb exploitation means that administrators have few active options for actually searching for the presence of logic bombs on their systems or systems introduced into their network. Many of the preventative measures for logic bombs are administrative in nature such as password management protocol and employee privilege monitoring [25]. While these administrative measures can make it more difficult for a malicious actor to introduce a logic bomb to the system, they provide nothing in the way of determining if a logic bomb is already on a system.

Most research into finding existing logic bombs depends on access to source code. There has been some promising research in the way of

static and dynamic code analysis to this end. These techniques seek to cover as many conditional branches in the code as possible to determine if there is a specific logic bomb condition. Most notably, branches that are very rarely reached or have extremely specific conditions in order to enter are the most suspicious.

SQA Tool, for example, was developed to detect logic bombs hidden within critical infrastructure software [2]. The basic premise of this tool is to create test conditions that SQA Tool then uses to maximize code coverage. The tool was able to achieve a high level of code coverage and ensure the potential malicious code was analyzed and removed from the critical infrastructure software before it was sent to customers.

Another group of researchers developed a tool called TriggerScope that has the sole goal of finding logic bombs within Android applications and removing the possibility for those applications ever reaching the Google Play Store [8]. This tool introduces a concept known as "trigger analysis" which is a static program analysis technique that finds the source of logic bomb trigger conditions [8]. TriggerScope was able to find previously unknown logic bombs that resided within applications on the Google Play Store and was demonstrated to help human analysts find and diagnose logic bombs [8].

Unfortunately, while both SQA Tool and TriggerScope have promising applications they are not directly useful for this research, as they rely on artifacts (source or Android APK) that we assume are not available.

2.4 Black Box Testing

The tools discussed in the previous section can greatly reduce the amount of time required to find a logic bomb within a piece of code. However, many commercial UAV platforms are not open source and require a different testing technique. Traditional software testing techniques fall within one of three categories: white box testing, gray box testing, and black box testing [14]. For this research, we are only concerned with black box testing. Black box testing simply means that the tester doesn't have access to any of the inner-workings of the device and only has knowledge of the inputs and outputs into the device/system [18]. Black box testing has many advantages over the previously mentioned software validation techniques as it is oftentimes faster on larger pieces of code, and the tester doesn't need to have intimate knowledge of the internal structure of the device or require knowledge of programming languages [14, 13].

Given the assumption that UAV autopilot source code is not available, the UAV must be treated as a black box. The detection method-

ology that has been developed loosely utilizes a black box testing technique known as "equivalence partitioning." Equivalence partitioning can be described as a method that divides input data into partitions and test cases are derived from those data partitions [13]. Through these means, the input space is greatly reduced while maximizing test conditions which creates a manageable input space that can be tested. Equivalence partitioning is achieved through using the flight profile for the specific mission set as a baseline for inputs rather than testing each combination of potential inputs within a mission area by brute force.

2.5 Summary

It is extremely important that critical infrastructure planners have the ability to validate that UAVs are able to perform their mission. As UAVs continue to fill roles of increasing responsibility, the effects of a compromised UAV could prove to be devastating. With previously documented cases of compromised UAVs, there is a very real threat to logic bombs lurking within UAV source code. Despite having proven logic bomb detection methods for other software, these solutions are unsuitable for this case given the user doesn't have access to the source code. For this reason, a new methodology for securing critical infrastructure UAVs must be developed.

3. Approach

3.1 Goals

The first and overall goal of this research is to develop a methodology that reduces the risk of a UAV encountering a logic bomb trigger event without having to brute force test all potential trigger conditions within a UAV operation. In order to achieve this, three sub goals are outlined. The first of these goals is to develop a simple logic bomb within the ArduPilot autopilot software. At the time of writing this article, we were unable to find any previous logic bomb that was developed for specifically ArduPilot autopilot software, or for UAVs in general. Secondly, it was essential that this logic bomb could be triggered and represented within the ArduPilot Software In the Loop (SITL) simulation software. This software serves as a key component of the suite used to validate the testing methodology. The third goal was to develop a log parsing/analysis program that can be utilized to perform analysis on the log files gathered from flights. This analysis defines a "safe corridor" through which a UAV can, with a high degree of confidence, travel without encountering a logic bomb trigger.

3.2 High Level Approach

The general approach taken toward this problem is to use the ArduPilot SITL simulator to execute real autopilot software through simulated missions prior to a real mission. These simulated missions aim to expose the autopilot software to realistic conditions that the UAV could reasonably expect to encounter during the course of its upcoming mission. By verifying that these conditions do not trigger a malicious logic bomb, mission planners can be confident that the UAV performs as expected in its actual mission. Additionally, by partitioning the flight plan and considering likely flight variations, "safe corridors" are developed through which the UAV can safely be expected to transit and operate.

Overall steps to the methodology are:

1 Partition flight according to expected type: point-to-point or free space

2 For point-to-point partitions, develop a confidence interval based "safe corridor" through which the UAV is expected, with specified confidence level, to stay

3 Within that corridor, exhaustively search all paths according to minimum resolution

4 For free space regions, exhaustively search all paths according to minimum resolution

5 If no logic bomb was triggered after exhaustive searches in parts 3 and 4, proceed with mission

3.3 Test Suite Overview

The following subsections will describe the test suite used for the experiment. This test suite consists of the ArduPilot SITL simulation software, MAVProxy Ground Control Software (GCS), Mission Planner GCS, and Python analysis code. These programs work in conjunction to define the test flight profiles, execute the flight profiles, and analyze the log files gathered from the flights. Figure 2 shows how the autopilot software interacts with the MAVProxy GCS as well as the Mission Planner GCS and shows how the autopilot receives input from the physics and flight engines to perform accurate testing.

As shown, the ArduPilot desktop executable interacts with the GCSs via TCP protocol over a serial connection and with the physics simulator via UDP connection. This simulation setup allows the autopilot software to interact with the physics engine.

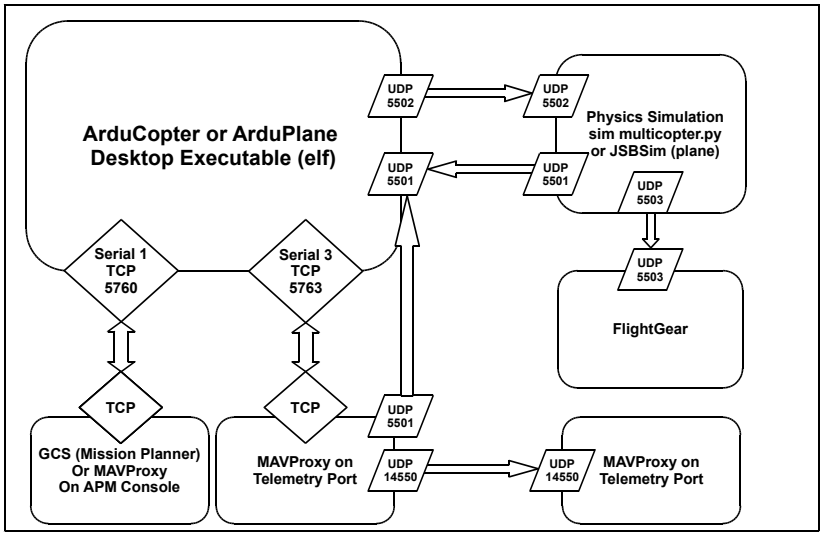


Figure 2: ArduPilot SITL Interaction[5].

3.3.1 ArduPilot SITL. ArduPilot is an open-source autopilot system that can be utilized on many types of vehicles. It provides advanced logging capabilities, simulation tools, and analysis tools [3]. ArduPilot has seen widespread use, having been installed on over a million vehicles. Perhaps the biggest strength of the ArduPilot software lies in the fact that the software is open source, allowing us to instrument the code with logic bomb malware for validating our methodology. Note that although ArduPilot is open source by nature, our methodology regards it as if it were closed source.

3.3.2 MAVProxy GCS. MAVProxy is a "fully-functioning GCS for UAVs" [16]. At its core, GCS software is responsible for sending and receiving commands/information to and from the UAV. Many ground control stations offer various other features and MAVProxy is no exception. MAVProxy provides various features such as a command-line control with basics graphical user interface capabilities, networking capabilities, portability, a light-weight design, and support for additional module loading and tab-completion for commands [16]. MAVProxy is the primary GCS utilized for these experiments and is the tool that is utilized to interface with the UAV and monitor flight path progress.

3.3.3 Mission Planner. Mission Planner is another GCS that was developed specifically for ArduPilot [4]. Mission Planner offers a few

advantages over MAVProxy with the main applications for this research being flight plan creation and logging capabilities. This GCS software allows for the easy creation of flight paths and gives the user the ability to save these flight plans for later use. When generating flight profiles, Mission Planner is essential to create the same flight path and to load the same flight data into the UAV before the flight begins.

3.3.4 Python Analysis Script. For this experiment, we are interested in looking for logic bombs that target the flight behavior of the UAV. The developed Python script parses out the log entries related to GPS information as well as waypoint information. In cases in which there was no detection of malicious activity, the data is grouped and organized by run, waypoint, and then by log entry. A confidence interval is then created across each run for each log entry on a one-to-one basis. This represents a "safe corridor" through which the UAV can travel while giving the planner high confidence that it will not encounter a logic bomb trigger.

3.4 Logic Bomb Creation

With the test suite defined and the basic log file analysis code completed, the next step in the process is to create a logic bomb within the ArduPilot autopilot code. The example logic bomb used for this research is designed to be easily identifiable upon activation. It targets the motors_update() function and prevents the motors from receiving new inputs from the autopilot once the trigger condition has been reached. The trigger condition was designed as a geofence, taking three parameters: altitude, latitude and longitude. With these parameters, a specific location in three dimensional space can serve as a trigger condition for the logic bomb. Upon activation of the trigger condition, the logic bomb causes the UAV to crash into the ground.

3.5 Test Missions

A series of four test missions are chosen that represent a basic array of real world missions: transit, circle, spline circle, and survey. These missions increase in level of complexity and the level of variance present within the flight path. On each of these missions, the ArduPilot autopilot desktop executable is complied with the logic bomb and its associated trigger condition(s). Note that all missions are carried out autonomously by the autopilot code besides arming, and performing the UAV takeoff to 5 meters.

3.5.1 Transit. The transit mission is designed to simulate a UAV taking off from a given location, transiting to a destination, and then landing. Although the logic bomb is present in all experiments, initially the geofence is set such that the UAV should not intersect it.

Variations in the ArduPilot SITL simulation weather and environmental factors mean that, just as in the real world, there will be slight variations in the actual path traveled by the UAV between set waypoints. The mission is executed for 10 runs under these conditions. The results are analyzed and a safety corridor is created within which the UAV can be expected to remain with 95% confidence. Using log entries, a confidence interval is generated in three dimensions along the entire flight path, accurately representing the predicted position of the UAV. The size of this corridor affects the next step, which is to exhaustively fly that corridor at some minimum resolution, or distance between flight tracks. For this validation, that minimum resolution was set to 1 meter due to the stated accuracy of GPS receivers fitted to typical small UAVs. Next, the geofence is configured such that the UAV will hit it in the course of its mission. This demonstrates the effectiveness of the logic bomb that has been created and creates a scenario that can be used as an example when developing the detection methodology.

3.5.2 Circle. The circle mission consists of takeoff, approach to the beginning of the circle holding area, flying a circular flight path, and landing. This type of flight path has various real world applications such as surveying a disaster area, or surveying critical infrastructure devices. As with the transit mission, the logic bomb trigger condition for the first 10 runs of the mission is set to be just outside the flight profile. A confidence interval is created across the runs to generate a flight corridor, and that flight corridor is exhaustively tested at minimum resolution. A final run is executed where the latitude, longitude, and altitude trigger conditions intersect the circular portion of the flight path and result in activation of the logic bomb.

3.5.3 Spline Circle. The spline circle mission is very similar to the circle mission with the difference being that during the spline circle mission, the altitude of the UAV increases with each subsequent lap around the circle. There are various real world applications for this mission type such as disaster area survey, or any other mission requiring the survey of an area from different altitudes. The same sequence of events takes place in regards to run and logic bomb activation. The primary advantage of this mission set is that it validates the circle area

for a wide variety of altitudes, providing the mission planner increased security assurance.

3.5.4 Free Area Survey. The survey mission is a bit different from the other test missions utilized in this experiment. This mission surveys an entire area of operation at a given altitude, assuring that there are no logic bomb triggers within that broad area. Using this mission set is the closest to a pure brute force approach but still significantly scales down the input set to only what the UAV is expected to experience. The UAV performs passes across an area with with 1 meter of separation, covering an entire area but still not having to input all potential inputs into the system to validate UAV operation within the area. Hence, the safety corridor is created in the entire region of the survey polygon, and a confidence interval is created only on the transit portions of the flight path. This solution gives operators the assurance that the UAV will be able to transit to the mission area and operate dynamically within the area at a given altitude.

3.5.5 Evaluation of Gathered Data. Much of the evaluation of the data that is gathered will be illustrated through simulation results in the form of examining the flight path log data and by the results generated from the analysis scripts. The results from those scripts create clear and defined areas of safe operation when a logic bomb isn't triggered by the ArduPilot SITL simulation. To identify programmatically when the logic bomb is triggered, flight data is examined for departures from the generated confidence interval. In such an event, it is assumed (and can be verified manually) that the UAV autopilot has suffered an event. These results will be examined in further depth in the analysis/discussion portion of this document.

4. Analysis of Experimental Results

Analysis of the results from the test flight paths are evaluated according to whether or not the experimental goals have been achieved, and by the measurable reduction in input space that is required for validate a mission flight path. Given that no other research could be found that addressed logic bomb detection with UAV autopilot code, it is assumed that the current best approach to finding logic bombs and their associated trigger conditions is a simple brute force technique. A brute force technique would have to evaluate all possible combinations of inputs in a given area of operation to a certain degree of precision to validate that a UAV will operate as expected. For this experiment, the level of precision for positional information is 1 meter based on the common GPS hardware that is deployed in small UAVs at this time. As the results show, this methodology greatly reduces the input space and time required to validate a mission set in a given location versus a pure brute force approach

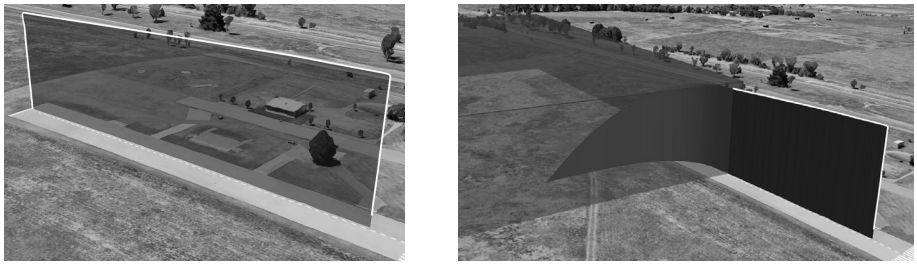
4.1 Logic Bomb Effectiveness

One of the derived goals of this research was to create a logic bomb specifically for the ArduPilot autopilot software and demonstrate its effectiveness within the ArduPilot SITL simulation software. This goal is vital because it not only proves that the a logic bomb can be executed within this autopilot code but that its operation can be accurately depicted within a simulated environment which can then be used as a test bed for this research. A logic bomb was developed within the ArduPilot autopilot code that has the ability to use latitude, longitude, and/or altitude parameters as trigger conditions for the logic bomb. While rudimentary, the logic bomb was extremely effective at disabling UAV motor operation and causing a loss of aircraft event (Figure 3b shows one example).

4.2 Transit Mission

As discussed, this flight path consists of takeoff, transit, and landing portions. Figures 3a and 3b show the output of these runs. The image on the right shows that once the logic bomb trigger area was reached, the UAV began to stray off course and eventually crashed.

The flight path figures clearly indicate that a logic bomb trigger condition has been reached. Shown in Figure 4 is sample output from the analysis script showing the confidence interval creation for the safety corridors for this mission set across 10 runs.



(a) Transit Mission With No Logic Bomb.

(b) Transit Mission With Logic Bomb Activation.

Figure 3: Transit Mission With and Without Logic Bomb Trigger Event.

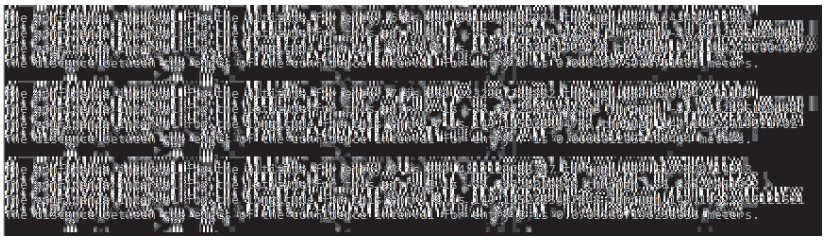
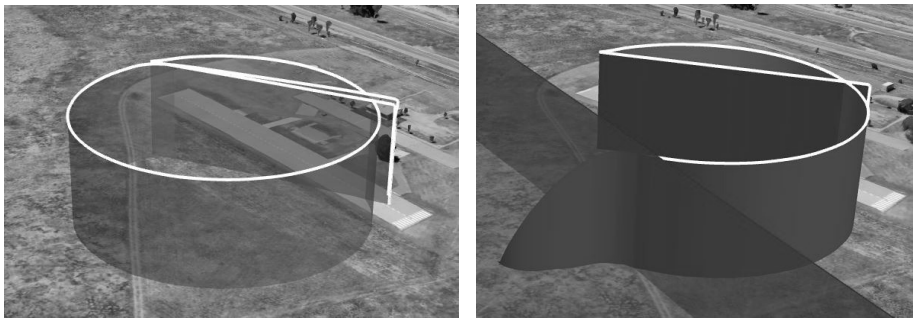


Figure 4: Sample Program Output.

Note the results show that the range on the generated confidence interval for safe UAV operation has little variance. This results from the ArduPilot SITL simulator. Although variances such as wind can be introduced, they have only a moderate effect on the flight of the UAV. A simulator that implemented more advanced weather effects would be needed to cause greater variation between runs. Because the confidence interval is less than the minimum resolution of 1 meter, the methodology does require a second step of exhaustively testing the safe corridor.

4.3 Circle Mission

The sequence of events for the circle mission are similar to that of the transit mission with the flight beginning with takeoff, a brief transit period to circle's edge, circle flight path, and finally landing. Figures 5a and 5b show this sequence of events without logic bomb activation as well as with logic bomb activation. Note that the figure on the left shows the mission plan execution from an angled view to clearly illustrate the different phases of flight while the figure on the right shows the flight path from the side view to illustrate the effect of the logic bomb.



(a) Circle Mission With No Logic Bomb.

(b) Circle Mission With Logic Bomb Activation.

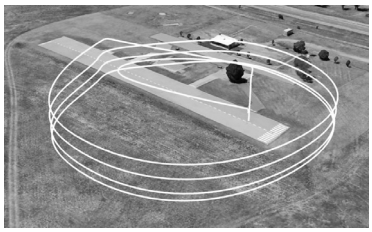
Figure 5: Circle Mission With and Without Logic Bomb Trigger Event.

The circular mission area has a diameter of $100m$ and area of approximately $7.9km^2$. To reduce complexity, it is assumed to be at a single altitude. Exhaustively searching then at a $1m$ resolution would require a simulated flight distance of approximately $7,854m$. By utilizing the flight path, the area of the search space is reduced to $314m$ per run (the circumference of the circle), with the UAV passing though all points of relevance on this mission. Using 10 runs to generate the confidence interval increases total simulated flight distance to $3,140m$, a reduction of 60%. The small variance observed between runs allowed for fewer runs to generate an acceptably small confidence interval in this case, but that may not always be true.

4.4 Spline Circle Mission

The spline circle mission is very similar to the circle mission except that multiple laps are taken around the circle. Instead of assuming one altitude as in the last use case, it is increased at a prescribed rate of 5 meters per lap. The logic bomb is set to include a specific altitude in the trigger condition. As illustrated in the experimental results, the UAV operates normally for multiple laps around the circle until a combined longitude, latitude, and altitude are reached. Figures 6a and 6b show the mission runs.

Unlike the circle mission that was only analyzed at one altitude, spline circle comparison accounts for 3 dimensions. The volume of the mission area extending from the ground to max altitude of 25m is $19.6m^3$. Exhaustively testing this full volume at a 1m lat/long resolution and 5m altitude resolution (for fair comparison) results in a simulated flight distance of $39.3km$. Each run developing a confidence interval requires only



(a) Spline Circle Mission With No Logic Bomb.

(b) Spline Circle Mission With Logic Bomb Activation

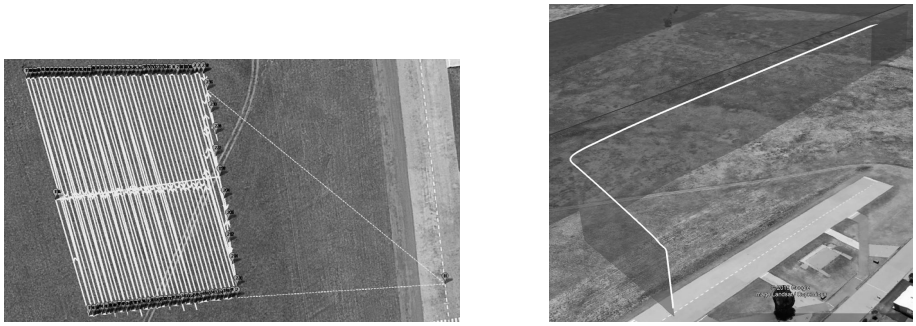
Figure 6: Spline Circle Mission With and Without Logic Bomb Trigger Event

$1.57km$, or a total of roughly $15.7km$ for 10 runs. As in the last case, this reduces the simulated flight time by a significant 60%.

4.5 Free Area Survey Mission

The survey mission is the closest to brute force detection methodology but can still provide significant benefit over simply brute forcing an entire area of operation when used in conjunction the reduction techniques described in the other cases. A confidence interval is generated only during the transit, takeoff, and landing portions of the mission. When the UAV reaches its intended "free" search area, the flight path calls for a sweeping pattern, with distances of 1 meter between sweep, of a very specific area of operation at a given altitude. Utilizing this technique, all theoretical coordinate information that could reliably activate a logic bomb within that area will be input, allowing for dynamic flying within an area during the real world mission. Figure 7a shows the survey flight path, both with and without logic bomb interference.

The volume of the area of operation in this flight path is $188,000m^3$. An area of this size leaves a huge number of inputs that must be validated. Utilizing 1 meter resolution, the UAV would have to traverse $188,000m$ of linear flight distance to validate the entire area of operation. Utilizing the takeoff, transit, and landing portions and then performing 1 meter sweeps in the defined area of operation, drastically reduces the amount of inputs that the UAV is required to experience to validate the mission. For example, instead of having to account for all inputs within a given country or township, the technique allows for the input space to be defined only to a given field or section of power line.



(a) Survey Mission With No Logic Bomb.

(b) Survey Mission With Logic Bomb Activation.

Figure 7: Survey Mission With and Without Logic Bomb Trigger Event.

4.6 Overall Analysis

This technique finds a solution for one of the critical problems when searching for logic bombs without having access to the source code; intractable input space. By utilizing the mission flight profile as a baseline for the input region, it has been demonstrated that the search space can be greatly reduced.

Generated safety corridors allow mission planners to safely plan real world mission routes with confidence that the UAV will not trigger an unknown logic bomb.

At this time, there was no other research discovered that searches for logic bombs hidden within UAV autopilot code, much less a methodology that does this without access to source code. For this reason, there are no other results to compare this methodology to besides the brute force approach. As discussed from the proof of concept experiments, this methodology performs significantly better than brute force input testing across an area of operation.

While these proof of concept experiments were designed to examine only latitude, longitude and altitude parameters, the Python analysis code can be modified to examine any loggable UAV parameter.

Confidence interval generation for other parameters can be completed alongside the latitude, longitude and altitude parameters which can be examined after simulated flight for potential logic bomb trigger condition activation.

5. Conclusion

The goal of this research is to create a logic bomb detection methodology for UAV autopilot systems that doesn't require access to the source code. With the growing importance of UAVs in the various critical infrastructure sectors, it is vital that these devices are secured before carrying out operations. While there are various other documented security issues, there has been little to no attention to the possibility that a logic bomb could be implanted within UAV autopilot code. By utilizing a simulation based logic bomb detection methodology, the key issues of finding logic bombs and their associated trigger events are minimized. This approach creates a manageable input space that can be searched by simply running the UAV through the proposed mission. Through this method, the search area is tailored specifically to the mission and entire areas of operation without the requirement to exhaustively search all input combinations. If the presence of a logic bomb isn't detected, mission planners are able to send the UAV on the mission without fear of unexpected behavior or the disastrous consequences associated with such behavior.

This paper has demonstrated that UAV flight paths can be validated before the execution of said mission. Additionally, this paper serves as a building block and a starting point for further research into logic bomb detection methodologies for UAVs. By adding more advanced features such as machine learning algorithms and gathering more log data information, this methodology has the potential to be robust at detecting and finding logic bombs that have even nuanced effects. Securing UAVs is critical for the future of critical infrastructure operations, and this research provides a starting point in an area that has received little attention to date.

References

[1] S. Adams and C. Friedland, A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management, *Ninth International Workshop on Remote Sensing for Disaster Response*, vol. 8, 2011.

[2] H. Agrawal, J. Alberi, L. Bahler, J. Micallef and A. Virodov, Detecting hidden logic bombs in critical infrastructure software, *Seventh International Conference on Information Warfare and Security*, 2012.

[3] ArduPilot, ArduPilot About (`ardupilot.org/about`).

[4] ArduPilot, Mission Planner Home (`ardupilot.org/planner/`).

[5] ArduPilot, SITL Simulator (Software in the Loop) (`ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html`).

[6] R. Austin, *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*, John Wiley and Sons, Chichester, West Sussex, United Kingdom, 2010.

[7] European Commission, Study Analysing the Current Activities in the Field of UAV, Enterprise and Industry Directorate-General, ENTR/2007/065, Brussels, Belgium (`ec.europa.eu/home-affairs/sites/homeaffairs/files/e-library/documents/policies/security/pdf/uav_study_element_2_en.pdf`), 2007.

[8] Y. Fratantonio, A. Bianchi, W. Robertson, E. Kirda, C. Kruegel and G. Vigna, TriggerScope: Towards detecting logic bombs in Android applications, *IEEE Symposium on Security and Privacy*, pp. 377–396, 2016.

[9] K. Hartmann and K. Giles, UAV exploitation: A new domain for cyber power, *Eighth International Conference on Cyber Conflict*, pp. 205–221, 2016.

[10] M. Hooper, Y. Tian, R. Zhou, B. Cao, A. Lauf, L. Watkins, W. Robinson and W. Alexis, Securing commercial WiFi-based UAVs from common security attacks, *IEEE Military Communications Conference*, pp. 1213–1218, 2016.

[11] A. Javaid, W. Sun, V. Devabhaktuni and M. Alam, Cyber security threat analysis and modeling of an unmanned aerial vehicle system, *Proceedings of the IEEE Conference on Technologies for Homeland Security*, pp. 585–590, 2012.

[12] R. Kandala, What are logic bombs and how to detect them, *Techulator* (`www.techulator.com/resources/`

10275-What-Logic-Bombs-How-detect-them.aspx), May 29, 2013.

[13] M. Khan, Different approaches to black box testing technique for finding errors, *International Journal of Software Engineering and Applications*, vol. 2(4), pp. 31–40, 2011.

[14] M. Khan and F. Khan, A comparative study of white box, black box and grey box testing techniques, *International Journal of Advanced Computer Science and Applications*, vol. 3(6), pp. 12–15, 2012.

[15] B. Madan, M. Banik and D. Bein, Securing unmanned autonomous systems from cyber threats, *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 16(2), pp. 119–136, 2019.

[16] MAVProxy, MAVProxy (ardupilot.github.io/MAVProxy/html/index.html).

[17] P. Mozur, Drone maker D.J.I. may be sending data to China, U.S. Officials Say, *New York Times* (www.nytimes.com/2017/11/29/technology/dji-china-data-drones.html), November 29, 2017.

[18] S. Nidhra and J. Dondeti, Black box and white box testing techniques - A literature review, *International Journal of Embedded Systems and Applications*, vol. 2(2), pp. 29–50, 2012.

[19] J. Pleban, R. Band and R. Creutzburg, Hacking and securing the AR.Drone 2.0 quadcopter: Investigations for improving the security of a toy, in *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications*, R. Creutzburg and D. Akopian (Eds.), SPIE, Bellingham, Washington, 2014.

[20] C. Rani, H. Modares, R. Sriram, D. Mikulski and F. Lewis, Security of unmanned aerial vehicle systems against cyber-physical attacks, *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 13(3), pp. 331–342, 2016.

[21] N. Rodday, R. Schmidt and A. Pras, Exploring security vulnerabilities of unmanned aerial vehicles, *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pp. 993–994, 2016.

[22] M. Smith, Leaked DHS memo accuses drone maker DJI of spying for China, *CSO* (www.csoonline.com/article/3239726/leaked-dhs-memo-accuses-drone-maker-dji-of-spying-for-china.html), December 3, 2017.

[23] J. Sullivan, Evolution or revolution? The rise of UAVs, *IEEE Technology and Society Magazine*, vol. 25(3), pp. 43–49, 2006.

[24] U.S. Department of Homeland Security, Critical Infrastructure Sectors, Washington, DC (www.dhs.gov/cisa/critical-infrastructure-sectors), 2019.

[25] K. Wehrum, Technology: When IT workers attack, *Inc.* (www.inc.com/magazine/20090401/technology-when-it-workers-attack.html), April 1, 2009.