An Uncrewed Aerial Vehicle Attack Scenario and Trustworthy Repair Architecture

Kate Highnam, Kevin Angstadt, Kevin Leach, Westley Weimer, Aaron Paulos[†], Patrick Hurley[‡]
University of Virginia [†]BBN Raytheon [‡]Air Force Research Laboratory
Charlottesville, VA 22904 Cambridge, MA 02138 Rome, NY 13441
{kwh5ye, angstadt, leach, weimer}@virginia.edu, apaulos@bbn.com, patrick.hurley.4@us.af.mil

***Abstract*—With the growing ubiquity of uncrewed aerial vehicles (UAVs), mitigating emergent threats in such systems has become increasingly important. In this short paper, we discuss an indicative class of UAVs and a potential attack scenario in which a benign UAV completing a mission can be compromised by a malicious attacker with an antenna and a commodity computer with open-source ground station software. We attest to the relevance of such a scenario for both enterprise and defense applications. We describe a system architecture for resiliency and trustworthiness in the face of these attacks. Our system is based on the quantitative assessment of trust from domain-specific telemetry data and the application of program repair techniques to UAV flight plans. We conclude with a discussion of restoring trust in post-repair UAV mission integrity.**

I. Introduction

The dramatic rise in the number of autonomous, uncrewed vehicles in use today has led to increased public discussion and scrutiny [36], and the promise of autopilot software operating aerial, terrestrial, and nautical vehicles has the potential to drastically reduce the risk of accidents. Cost-effective and safe surveillance, transport, and military applications are thought to become primary reasons to employ autonomous vehicles [22]. This has sparked research concerning the safety of deploying uncrewed vehicles in situations with possible human and material costs [22]. The DARPA High-Assurance Cyber Military Systems program [18], which produced formally-hardened embedded systems, elevates the bar by making such vehicles more difficult to hack [26]. Nevertheless, we believe that resiliency is still critical, especially when systems are inevitably used outside of their anticipated operating environment. For example, Google's self-driving car, touted for its safety [17], recently had its first reported crash [9]; deploying such a vehicle in a trustworthy manner remains a challenge. We must develop notions of dependability and trustworthiness for the safety and resiliency of autonomous vehicles before their successful large-scale deployment in industrial and military applications.

For this paper and in the context of UAVs, *dependability* is a measure of how consistently the platform successfully completes its assigned mission (a list of tasks given to a UAV to fulfill) [8]. Such a system is *trustworthy* if the human operators believe it to be dependable, demonstrated through the UAV's reliability in continuing to follow the correct mission plan loaded at takeoff and maintaining the integrity of the mission specifications throughout a safe completion. Environmental, human, software, and hardware factors can all adversely affect the vehicle's ability to perform a given mission. A *resilient* system is capable of safely recovering from or avoiding these hindrances to complete the original mission or a variation thereof, restoring integrity to the system [14].

In this paper, we identify and motivate a realistic scenario (possible today) in which 1) an autonomous vehicle is compromised by an attacker through a software vulnerability; 2) the attack is detected as it is occurring; 3) a repair is deployed that compensates for the attack, allowing the vehicle to complete its mission; and 4) artifacts associated with the system and the repair are used to re-establish trust in the post-repair system.

Our specimen scenario involves a UAV that is programmed to complete a mission involving visiting a sequence of waypoints and photographing a valuable landmark. Such a mission contains components of indicative importance to industrial [15] and defense [5] applications. We assume trust in the system before the attack occurs. A malicious agent launches an attack that prevents mission completion by altering the flight plan such that the vehicle never reaches certain waypoints, resulting in a loss of trust in the system. This attack scenario is possible now using commodity hardware and software [6]. Our system detects the attack as it occurs and automatically synthesizes a new, repaired mission plan that mitigates the threat. After deployment, the vehicle can complete its mission successfully despite the attack. Through analyses of pre- and post-repair information we can re-establish trust in the vehicle's ability to complete the mission.

We consider two commodity autonomous aerial vehicles and focus on an attack scenario in which an attacker has an antenna capable of transmitting malicious data that compromises the flight software. We discuss how we measure our notion of trust and how the attack in our scenario reflects a loss of trust. Finally, we describe an architecture in which we can automatically detect such an attack, synthesize a repaired mission mid-flight, deploy and complete the repaired mission, and reevaluate and reestablish trust in the UAV.

II. Background

First, we briefly consider related work associated with autonomous vehicle dependability, trust, and resilient systems.

Acknowledgment Of Support And Disclaimer: (a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work funded by AFRL, under AFRL Contract No. FA8750-15-2-0075. (b) Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL. Approved for Public Release, Unlimited Distribution: 88ABW-2016-1226, 3/15/2016.

a) Autonomous Vehicle Dependability: In recent years, several high-profile accidents involving autonomous vehicles have been brought to light [13], [33]–[35]. These incidents motivate increased system dependability. Researchers have improved hardware resiliency [18], [24], [26], but mission and software resiliency remain open problems.

b) Trusting Resilient Systems: Recent research efforts in resilient systems have demonstrated an ability to automatically repair deployed software to fight through failures and attacks include both search-based [16], [19], [28], and formal approaches [23], [25]. Search-based and evolutionary techniques have proven to be highly resilient but less trusted while formal approaches have proven to be highly trusted but less resilient. For example, the recent Angelix approach to program repair, based on formal methods and constraint solving, produces repairs for 28 out of 82 defects considered [23, Tab. 2], while the search-based GenProg algorithm produces repairs for 48 of those 82 [19, Tab. IV]. Not all candidate repairs are equally trusted by all clients (e.g., [12], [16], [21]). The DARPA CRASH program had similar resiliency goals but was not targeted to embedded or legacy systems. We consider a system architecture in which either search-based or formal approaches (e.g., [29], [30]) may be used to produce repairs, and additional post-hoc analyses are used to re-establish trust.

III. Specimen Systems and Attack Scenario

The commodity UAVs (quadcopters) we consider run open-source autopilot software [2] atop Unix-like operating systems with real-time kernels. These vehicles are controlled remotely via the Micro Autonomous Vehicle Link (MAVLink) protocol [3]. MAVLink is a packet-based communication protocol used for data exchange between ground station software (e.g., APMPlanner [1] or QGroundControl [4]) and the UAV, including motion commands and telemetry information. Physical and link-layer communication is achieved via radio telemetry devices. Communication is unencrypted and relies on a System ID number to distinguish multiple vehicles. Attackers can capture the System ID of an in-flight UAV and spoof MAVLink packets to control that vehicle. Such an attack has already been publicly demonstrated by hobbyists using only $25 worth of hardware [6]. This high-level UAV architecture is indicative of popular commodity offerings such as the 3DR IRIS$^+$ [7], Erle-Copter [11], and various Raspberry Pi kit copters [10].

We assume an uncrewed vehicle on a surveillance patrol mission; the mission objectives include aiming an onboard camera at each of a series of fixed waypoints during patrol. While presented here abstractly, both the UAV properties and the mission described here are also indicative of deployments that are of commercial [15] or defense [5] interest.

Figure 1a illustrates the specimen attack scenario considered in this paper. An attacker with ground station software and a directed antenna sends malicious MAVLink packets repeatedly. The attacker's signal radiates in a cone pattern (shown in Figure 1a as the *Cone of influence*). Upon vehicle entry into the cone of influence, the attacker can send packets that divert the path or heading of the vehicle (e.g., spoofed `SET_POSITION_TARGET_GLOBAL_INT` packets), preventing capturing of a focused picture of the waypoint. We assume that the attacker is motivated by subtlety and will *not* seek to simply crash the vehicle or otherwise overtly draw attention. Such an attacker seeks to slightly influence the vehicle, concealing the exact source location of the attack.

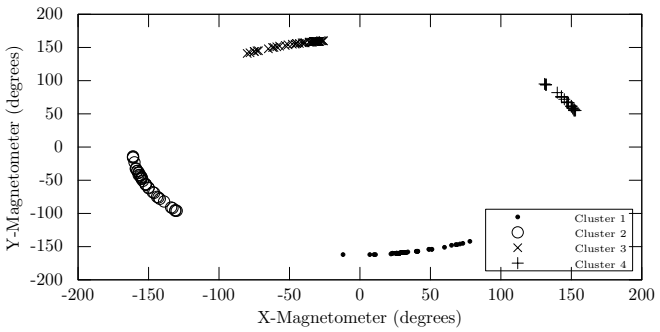

Fig. 2: Clusters of X and Y magnetometer telemetry.

A possible solution is an alternate version of MAVLink supporting encryption and authentication; however, in the present and foreseeable future, this is not viable. First, it requires additional processing and battery power, both on the UAV (where there is significant pressure for smaller vehicles) and in manual controllers (where there is significant cost pressure). We predict that commodity devices will continue to be sold and deployed without it. Second, encryption does not always defeat all classes of capture-replay attacks [27].Third, even if packets are encrypted, UAVs are still potentially vulnerable to other security attack vectors, such as buffer overruns in packet headers or return-oriented programming (ROP) attacks.

IV. Trustworthy Repair Architecture

In this section, we present a high-level architecture for the dependable and trustworthy completion of the surveillance mission in the face of the stealthy cone of influence attack. We consider dependability (i.e., consistent completion of missions) and trustworthiness (i.e., operator trust in the post-attack, post-repair behavior of the system) separately. First, we detect the attack using collected telemetry values and a suitable software framework. Second, we programmatically develop a satisfactory repair that allows successful completion of the mission despite the attack. Finally, we deploy the repair. Our overall architecture is shown in Figure 1b.

A. Quantitative Trust and Attack Detection

We propose an empirical approach that combines UAV and mission profiling along with runtime tools to quantitatively and continuously measure progress against a defined mission. We apply flight simulation, light-weight statistics and data clustering to externally profile telemetry data at the ground station. Telemetry data provides a rich set of inter-related observations about sensed values, actuator set-points, and composite application states. We hand-selected basic telemetry messages related to position, heading, speed, and consumed

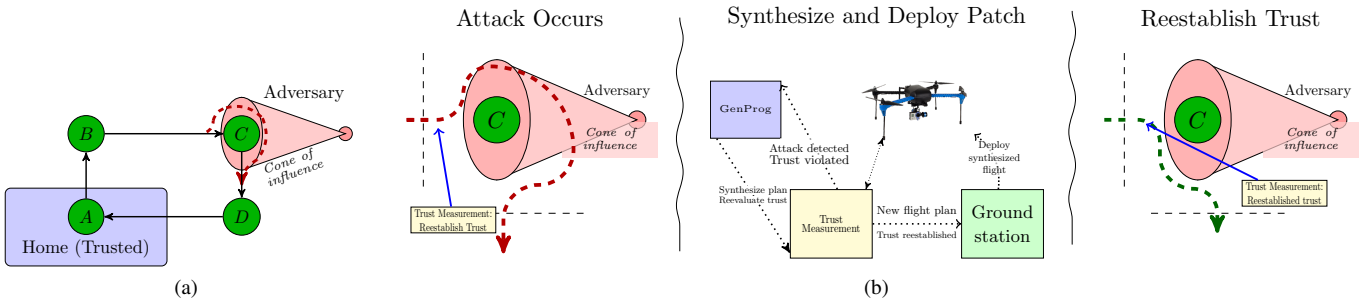

Fig. 1: (a) Proposed attack scenario. A UAV visits waypoints A–D via the desired flight plan (black arrows). An attacker causes the vehicle to traverse the dashed dark red path, causing waypoint C to be missed and the mission to fail. (b) Proposed architecture for trustworthy resilient operation. By continually measuring mission trustworthiness, we can detect the attack as it occurs. We use automated repair to synthesize and deploy a new flight plan to avoid the attacker's antenna.

resources (e.g., time taken and battery usage). Through simulation and telemetry profiling, we are able to extract mission constraints describing how a vehicle should move between waypoints and what type of telemetry it should produce. An example of such a constraint is shown in Figure 2, where X- and Y-magnetometer samples, from -180 to 180 degrees, are clustered for a square surveillance mission. The raw readings for the two samples are naturally partitioned into four clusters that define bearing through four legs of a square mission.

We postulate that it is possible to detect some attack effects by examining inter-related and/or redundant telemetry data. Consider an attack causing the UAV to fly off course and falsify GPS data, including latitude, longitude, and course-of-ground estimates. In this case, while a direct runtime assessment would most likely confirm that the spoofed GPS data fits within the mission execution profile, by examining an inter-related set of observations, there is an increased likelihood of detecting the spoofed GPS data. This would be achieved by cross-checking the raw magnetometer readings against the clusters found in Figure 2.

B. Repair Synthesis

We next synthesize an altered flight plan to complete the desired mission while avoiding the cone of influence. We propose to use an off-the-shelf automated program repair technique, such as GenProg [19], to do so. GenProg is an evolutionary approach that is inherently resilient due to its ability to evolve a solution but provides limited trust guarantees since it uses a stochastic approach to fight through attacks. GenProg searches through the space of possible repairs, returning one that maximizes an objective function. Adapting program repair to this scenario requires two key design decisions: the representation of a flight plan and the construction of a function that assesses candidate flight plan quality with respect to the mission.

We consider two flight plan representations. First, the plan may be represented as a sequence of raw, serialized MAVLink packets that would be sent to the UAV upon initialization. GenProg has demonstrated success on repairs to similar substrates, such as Executable and Linking Format (ELF) binaries (which can also be viewed as a container format holding uninterpreted data) and has been shown to work quite well in embedded systems [31]. Second, the plan may be represented as a C program with calls to MAVLink library functions. A programmatic representation not only allows for dynamic flight plans (i.e., those that change on-the-fly based on observations or variables) but also fits directly with an expressive set of tree-structured edits used by GenProg to explore control- and data-flow alterations [20]. Other representations, such as those based on formal geometry, are possible but remain future work.

Given a representation, automated program repair approaches produce several candidate flight plans for evaluation. Existing simulation software accepts both representations (raw packets and programmatic control) and produces traces of telemetry data. We propose an objective function that analyzes this trace. For example, the objective function could return the number of waypoints successfully surveilled, with penalties for remaining near the cone of influence. Program repair also supports multiple objectives [32], allowing for additional considerations (e.g., flight duration, power usage, etc.).

C. Repair Deployment

Repair deployment depends on the flight plan representation used during synthesis. For a serialized list of MAVLink packets, we instruct the UAV to hover in place in a safe region. We then send the new flight plan (list of MAVLink packets) to the vehicle, and switch back into autonomous mode, allowing the UAV to continue the mission at the nearest waypoint. For a mission representated as a C program, deployment consists of running the newly-synthesized program.

V. Toward Trustworthy UAV Resilience

As part of the repair synthesis process, we have a quantitative notion of the repair's quality based on simulated telemetry information. After repair deployment, we assess repair quality based on *measured* telemetry information. If the measured information differs significantly (e.g., because of changing environmental conditions or an unsimulated repair consequence), another repair can be constructed. Otherwise, the measured information can be compared to the historical training data to assess trust in post-repair systems. The pre-

and post-repair system are expected to differ in certain respects (e.g., avoidance of the cone of influence), but those are known statically during repair synthesis and can be communicated to the trust assessment system. Evidence of trust in the post-repair system can thus be mechanically provided with respect to statistics that should not have changed (e.g., maximum flight speed), trust in other elements assessed manually (e.g., positions during flight), and the knowledge of the overall mission success (e.g., number of waypoints surveilled). While this quantitative notion of trust only addresses some measured aspects of behavior, the current state of the art does not involve any explicit information about the post-repair system.

Despite several federally-funded efforts, it is still difficult to determine whether a system is trustworthy. Evaluating the trustworthiness of a system throughout mission execution—including after resiliency actions—is similarly challenging. While our proposed architecture is only a preliminary step toward answering such questions, it does provide a point of comparison and serve as a baseline for such a system.

VI. Conclusions and Future Work

The growing interest in UAVs, combined with documented system vulnerabilities, necessitates new solutions in system resiliency. In this short paper we describe a simple mission (waypoint surveillance) and attest to its industrial and defense relevance. We consider an attack scenario, the stealthy cone of influence, in which an attacker can divert the flight plan of an airborne vehicle, causing the mission to fail; we attest to its relevance as well. We then describe a framework for resiliently addressing this attack and then re-establishing trust in the resulting system. The two key components of our architecture are a quantitative assessment of trust based on models of domain-specific telemetry information and the application of automated repair techniques to flight plans. Ultimately, we argue for a quantified measure of trust to provide the human operator with confidence that the mission is trustworthy and will succeed following patch deployment. More general patch synthesis and trust metrics remain open problems.

We hope this indicative mission and attack scenario will motivate and crystallize subsequent research, and that our proposed architecture can serve as a baseline for future comparison. Now is the right time to take on these research challenges. Adaptive response is widely accepted as a necessity to operate through attacks, but system owners and operators may be reluctant to use this new technology because it lacks the advances required to earn their trust. Past and ongoing research in trust, dependability, and resilience present an opportunity to develop a foundation for trusted, resilient UAVs.

References

[1] "APM mission planner 2," http://planner2.ardupilot.com.

[2] "ArduPilot," http://ardupilot.com.

[3] "MAVLink micro air vehicle communication protocol," http://qgroundcontrol.org/mavlink/start.

[4] "QGroundControl," http://qgroundcontrol.org/start.

[5] "Unmanned aircraft systems (UAS)," *Department of Defense Purpose and Operational Use*, 2006.

[6] "Hijacking drones with a MAVLink exploit," *DIY Drones*, 2015.

[7] 3DRobotics, "3DR IRIS$^+$," https://store.3dr.com/products/iris.

[8] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[9] C. DMV, "Report of traffic accident involving an autonomous vehicle," *California Department of Motor Vehicles*, February 2016.

[10] Emlid, "Navio2: Linux autopilot on Raspberry Pi," http://www.emlid.com.

[11] Erle Robotics, "Erle-Copter," http://erlerobotics.com/blog/erle-copter/.

[12] Z. P. Fry, B. Landau, and W. Weimer, "A human study of patch maintainability," in *International Symposium on Software Testing and Analysis*, 2012, pp. 177–187.

[13] K. Gander, "Drone delivering asparagus to Dutch restaurant crashes and burns," *The Independent*, March 2015.

[14] Y. Y. Haimes, "On the definition of resilience in systems," *Risk Analysis*, vol. 29, no. 4, pp. 498–501, November 2009.

[15] B. Handwerk, "5 surprising drone uses (besides Amazon delivery)," *National Geographic*, December 2013.

[16] D. Kim, J. Nam, J. Song, and S. Kim, "Automatic patch generation learned from human-written patches," in *International Conference on Sofware Engineering*, 2013.

[17] W. Knight, "Google's self-driving-car chief defends safety record," *MIT Technology Review*, July 2015.

[18] J. Launchbury, "High-assurance cyber military systems (HACMS)," *DARPA Program Information*, November 2015.

[19] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer, "A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each," in *International Conference on Software Engineering*, 2012.

[20] C. Le Goues, S. Forrest, and W. Weimer, "Representations and operators for improving evolutionary software repair," in *Genetic and Evoluationary Computation Conference*, 2012, pp. 959–966.

[21] F. Long and M. Rinard, "Staged program repair with condition synthesis," in *Foundations of Software Engineering*, 2015, pp. 166–178.

[22] T. Lozano-Perez, I. J. Cox, and G. T. Wilfong, *Autonomous robot vehicles*. Springer Science & Business Media, 2012.

[23] S. Mechtaev, J. Yi, and A. Roychoudhury, "Angelix: Scalable multiline program patch synthesis via symbolic analysis," in *International Conference on Software Engineering*, 2016 (To appear).

[24] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *Robotics and Automation*, May 2014, pp. 45–52.

[25] H. D. T. Nguyen, D. Qi, A. Roychoudhury, and S. Chandra, "SemFix: Program repair via semantic analysis," in *International Conference on Sofware Engineering*, 2013, pp. 772–781.

[26] P. Paganini, "Hack-proof drones possible with HACMS technology," http://resources.infosecinstitute.com/hack-proof-drones-possible-hacms-technology/, 2014.

[27] R. Pries, W. Yu, X. Fu, and W. Zhao, "A new replay attack against anonymous communication networks," in *International Conference on Communications*. IEEE, 2008, pp. 1578–1582.

[28] Y. Qi, X. Mao, Y. Lei, Z. Dai, and C. Wang, "The strength of random search on automated program repair," in *International Conference on Software Engineering*, 2014.

[29] R. v. R. Robert Constable, Mark Bickford, "Investigating correct-by-construction attack-tolerant systems," Cornell University, Tech. Rep., 2010. [Online]. Available: http://hdl.handle.net/1813/23575

[30] N. Schiper, V. Rahli, R. V. Renesse, M. Bickford, and R. L. Constable, "Developing correctly replicated databases using formal tools," in *Systems and Networks*, 2014, pp. 395–406.

[31] E. Schulte, J. DiLorenzo, S. Forrest, and W. Weimer, "Automated repair of binary and assembly programs for cooperating embedded devices," in *Architectural Support for Programming Languages and Operating Systems*, 2013.

[32] E. Schulte, J. Dorn, S. Harding, S. Forrest, and W. Weimer, "Post-compiler software optimization for reducing energy," in *Architectural Support for Programming Languages and Operating Systems*, 2014.

[33] G. Smith, "Why drone enthusiasts all over the country are getting arrested," *The Huffington Post*, July 2014.

[34] R. R. Sobol, "Popular hobby drones pose danger when drivers lose control," *The Chicago Tribune*, January 2015.

[35] J. Spector, "The future of hobby drone regulation is up in the air," *CityLab, The Atlantic*, August 2015.

[36] H. Sreenivasan, "Rise of domestic drones draws questions about privacy, limiting use," *PBS News Hour*, April 2013.