R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems*

Johann Schumann[1], Patrick Moosbrugger[1], and Kristin Y. Rozier[2]

[1] SGT, Inc., NASA Ames, Moffett Field, CA, USA, Johann.M.Schumann@nasa.gov, Patrick.Moosbrugger@technikum-wien.at
[2] University of Cincinnati, OH, USA, Kristin.Y.Rozier@uc.edu

Abstract. We present R2U2, a novel framework for runtime monitoring of security properties and diagnosing of security threats on-board Unmanned Aerial Systems (UAS). R2U2, implemented in FPGA hardware, is a real-time, REALIZABLE, RESPONSIVE, UNOBTRUSIVE Unit for security threat detection. R2U2 is designed to continuously monitor inputs from the GPS and the ground control station, sensor readings, actuator outputs, and flight software status. By simultaneously monitoring and performing statistical reasoning, attack patterns and post-attack discrepancies in the UAS behavior can be detected. R2U2 uses runtime observer pairs for linear and metric temporal logics for property monitoring and Bayesian networks for diagnosis of security threats. We discuss the design and implementation that now enables R2U2 to handle security threats and present simulation results of several attack scenarios on the NASA DragonEye UAS.

1 Introduction

Unmanned Aerial Systems (UAS) are starting to permeate many areas in everyday life. From toy quadcopters, to industrial aircraft for delivery, crop dusting, public safety, and military operations, UAS of vastly different weight, size, and complexity are used. Although the hardware technology has significantly advanced in the past years, there are still considerable issues to be solved before UAS can be used safely. Perhaps the biggest concern is the integration of UAS into the national airspace (NAS), where they have to seamlessly blend into the crowded skies and obey Air Traffic Control commands without endangering other aircraft or lives and property on the ground [5].

A related topic, which has been vastly neglected so far, is security [24]. All sensors and software set up to ensure UAS safety are useless if a malicious attack can cause the UAS to crash, be abducted, or cause severe damage or loss of life. Often, live video feeds from military UAS are not encrypted, so people on the ground, with only minimal and off-the-shelf components, could see the same images as the remote UAS operator [30]. In 2011, Iran allegedly abducted a CIA drone by jamming its command link and spoofing its GPS. Instead of returning to the CIA base, the UAS was directed to land on Iranian territory [6]. Even large research UAS worth millions of dollars are controlled via unencrypted RF connections; most UAS communicate over a large number of possible channels [9], relying on the assumption that "one would have to know the frequencies" to send and receive data.

There are multiple reasons for these gaping security holes: most UAS flight computers are extremely weak with respect to computing power. Thus, on-board encryption

* This work was supported in part by NASA ARMD 2014 I3AMT Seedling Phase I, NNX12AK33A and the Austrian Josef Ressel Center (VECS).

is not possible, especially for larger data volumes as produced, for example, by on-board cameras. Another reason is that a lot of UAS technology stems from the Hobby RC area, where security is of low concern. Finally, security aspects have only played a minor role in FAA regulation to date [7].

On a UAS, there are multiple attack surfaces: the communication link, sensor jamming or spoofing, exploitation of software-related issues, and physical attacks like catching a UAS in a net. In this paper, we focus on the detection of communication, sensor, and software-related security threats, but do not elaborate on attack prevention or possible mitigation strategies. Though design-time verification and validation activities can secure a number of attack surfaces, an actual attack will, most likely, happen while the UAS is in the air. We therefore propose the use of dynamic monitoring, threat detection, and security diagnosis.

In order to minimize impact on the flight software and the usually weak flight computer, R2U2 is implemented using FPGA hardware. This no-overhead implementation is designed to uphold the FAA requirements of REALIZABILITY and UNOBTRUSIVENESS. To our knowledge, there are only two previous embedded hardware monitoring frameworks capable of analyzing formal properties: P2V [15] and BusMOP [23,19]. However, P2V is a PSL to Verilog compiler that violates our UNOBTRUSIVENESS requirement by instrumenting software. Like R2U2, BusMOP can monitor COTS peripherals, achieving zero runtime overhead via a bus-interface and an implementation on a reconfigurable FPGA. However, BusMOP violates our REALIZABILITY requirement by reporting only property failure and handling only past-time logics whereas we require early-as-possible reporting of future-time temporal properties passing and intermediate status updates. BusMOP also violates UNOBTRUSIVENESS by executing arbitrary user-supplied code on the occurrence of any property violation.

Previously, we developed our on-board monitoring and diagnosis framework R2U2 for system health management of hardware-only components and developed implementations to detect hardware failures [26,8,28]. We defined and proved correct our FPGA temporal logic observer encodings [26] and our Bayesian network (BN) encodings [8], which comprise R2U2's underlying health model. We also envisioned a compositional building-block framework for integration with other diagnosis technologies that also analyzed software components [28]; in this paper, we follow up on that idea by providing the first implementation of R2U2 that includes software components.

Here, we extend R2U2 to enable the dynamic monitoring of the flight software, the communication stream, and sensor values for indications of a malicious attack on the autopilot and, even more importantly, to be able to quickly and reliably detect post-attack behavior of the UAS. The temporal and probabilistic health models and their FPGA implementations are suited for fast detection and diagnosis of attacks and post-attack behavior. The separate FPGA implementation of a security extension to R2U2 described in this paper is highly resilient to attacks, being an isolated hardware entity and programmed using VHDL. Javaid et al. [10] also analyze cybersecurity threats for UAS. They simulated the effects of attacks that usually ended in a crash, focusing on identifying different existing attack surfaces and vulnerabilities rather than focusing on runtime detection or post-attack analysis. TeStID [2], ORCHIDS [21] and MONID [20] are intrusion detection systems that use temporal logic to specify attack patterns. These

security monitoring frameworks are targeted at IT systems and infrastructure.
Our contributions include:

- extending R2U2 from monitoring of safety properties of hardware [26,8] to integrating hardware and software bus traffic monitoring for security threats thus enabling on-board, real-time detection of attack scenarios and post-attack behavior;
- detection of *attack patterns* rather than component failures;
- ensuring monitoring and reasoning are isolated from in-flight attacks; our FPGA implementation provides a platform for secure and independent monitoring and diagnosis that is not re-programmable in-flight by attackers;
- demonstrating R2U2 via case studies on a real NASA DragonEye UAS; and
- implementing a novel extension of R2U2 that we release to enable others to reproduce and build upon our work: `http://temporallogic.org/research/RV15.html`.

The rest of this paper is structured as follows. Section 2 provides background information on our UAS platform, the open-source flight software, and the R2U2 framework. Section 3 is devoted to our approach of using temporal logic observers and BN diagnostic reasoning for detection of security threats and post-attack UAS behavior. In Section 4, we will illustrate our approach with several small case studies on attacks through the ground control station (GCS), attempts to hijack a UAS through an attacker GCS, and GPS spoofing. Finally, Section 5 discusses future work and concludes.

2 Background

For this paper, we consider a simple and small UAS platform, the NASA DragonEye (Figure 1A). With a wingspan of 1.1m it is small, but shares many commonalities with larger and more complex UAS. Figure 1B shows a high-level, generic UAS architecture: the UAS is controlled by an on-board flight computer running the flight software (FSW). It receives measurements from various sensors, like barometric pressure and airspeed, GPS, compass readings, and readings from the inertial measurement unit (IMU). Based upon this information and a flight plan, the FSW calculates the necessary adjustments of the actuators: elevator, rudder, ailerons, throttle. A ground control station (GCS) computer transmits commands and flight plans to the UAS, and receives and displays UAS telemetry information. For fully autonomous missions, there is no link between the UAS and the GCS.

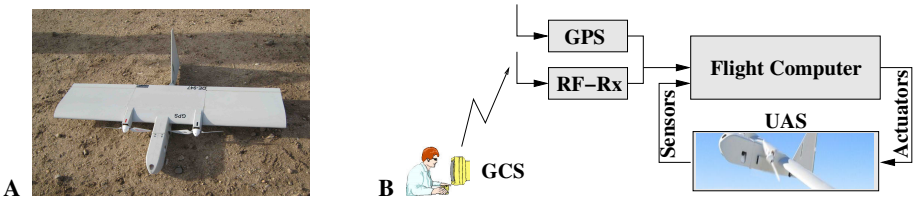

Fig. 1. A: Photo of NASA DragonEye. B: High level system architecture of a small UAS.

Our example system uses the open-source FSW ”APM:Plane” [3], which does not contain any security features like command or data encryption for the GCS-UAS link

per default. We nevertheless selected this FSW because it very closely resembles the architecture of both similarly small and larger, more complex UAS. This architecture allows us to easily carry out white-box experiments and to study the relationship between attacks and post-attack behavior. Results of our studies can be carried over to highly secure and resilient flight software.

2.1 R2U2

Developed to continuously monitor system and safety properties of an UAS in flight, our real-time R2U2 (REALIZABLE, RESPONSIVE, and UNOBTRUSIVE Unit) has been implemented on an FPGA (Field Programmable Gate Array). Hierarchical and modular models within this framework are defined using Metric Temporal Logic (MTL) and mission-time Linear Temporal Logic (LTL) [26] for expressing temporal properties and Bayesian Networks (BN) for probabilistic and diagnostic reasoning. In the following, we give a high-level overview over the R2U2 framework and its FPGA implementation. For details on temporal reasoning, its implementation, and semantics the reader is referred to [26]; [8] describes details on the FPGA implementation of Bayesian networks. Also [29] provides details on R2U2 modeling and system health management.

Temporal Logic Observers LTL and MTL formulas consist of propositional variables, the logic operators $\wedge$, $\vee$, $\neg$, or $\rightarrow$, and temporal operators to express temporal relationships between events. For LTL formulas p, q, we have $\Box p$ (ALWAYS p), $\Diamond p$ (EVENTUALLY p), $\mathcal{X} p$ (NEXTTIME p), $p \mathcal{U} q$ (p UNTIL q), and $p \mathcal{R} q$ (p RELEASES q) with their usual semantics [26]. For MTL, each of the temporal operators are accompanied by upper and lower time bounds that express the time period during which the operator must hold. Specifically, MTL includes the operators $\Box_{[i,j]}\ p$, $\Diamond_{[i,j]}\ p$, $p\, \mathcal{U}_{[i,j]}\ q$, and $p\ \mathcal{R}_{[i,j]}\ q$ where the temporal operator applies over the interval between time i and time j, inclusive, and time steps refer to ticks of the system clock. For mission-bounded LTL operators these time bounds are implied to be the start and end of the UAS mission.

Bayesian Networks for Health Models In many situations, temporal logic monitoring might find several violations of security and safety properties. For example, a certain system state might have been caused by an attack or by a bad sensor; we can use the combination of property violations to determine which one. In order to be able to disambiguate the root causes, the R2U2 framework uses Bayesian Networks (BN) for diagnostic reasoning. BNs are directed acyclic graphs, where each node represents a statistical variable. They are well-established in the area of diagnostic and health management (e.g., [22,18]). Conditional dependencies between the different statistical variables are represented by directed edges; local conditional probabilities are stored in the Conditional Probability Table (CPT) of each node [8,27,29]. R2U2 evaluates posterior probabilities, which reflect the most likely root causes at each time step.

2.2 FPGA Implementation

R2U2 is implemented in FPGA hardware (Figure 2). Signals from the flight computer and communication buses are filtered and discretized in the signal processing (SP) unit to obtain streams of propositional variables. The runtime verification (RV) and runtime

reasoning (RR) units comprise the proper health management hardware: the RV unit monitors MTL/LTL properties using pairs of synchronous and asynchronous observers defined in [26]. After the temporal logic formulas have been evaluated, the results are transferred to the RR subsystem, where the compiled Bayesian network is evaluated to yield the posterior marginals of the health model.

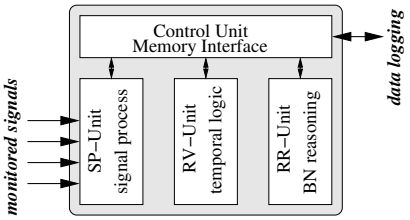

Fig. 2. Principled R2U2 implementation

3 Our Approach to Threat-Detection

For our approach, we consider the "system" UAS (as depicted in Figure 1B) as a complex feedback system. Commands, GPS readings, and measurements of the sensors are processed by the FSW on the flight computer to calculate new values for the actuators, and to update its internal status. In this paper, we assume that all malicious attacks are attempted via communication during flight.[3] Furthermore, all communications to the UAS are received via a wireless link from the ground control station, or GPS satellites, or transmitters only. Spoofing of the compass sensor, for example, via a strong magnetic field is outside the scope of R2U2.

With our R2U2 framework, we continuously monitor inputs from ground control and GPS and can identify many attack mechanisms and surfaces. Typical examples include denial-of-service, sending of illegal or dangerous commands, or jamming of the GPS receiver. Because, in most cases, information about the communication does not suffice to reliably identify an attack scenario, additional supporting information is necessary. This will be obtained from the analysis of post-attack behavior of the UAS. Any successful attack on the UAS will result in some unusual and undesired behavior of the UAS.

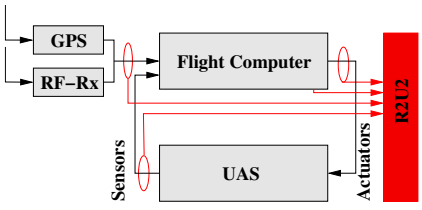

Fig. 3. High-level architecture of R2U2

Monitoring the system inputs and analyzing the post-attack behavior are not independent from each other so we have to model their interactions within our R2U2 framework. Typically, a certain input pattern followed by a specific behavior characterizes an attack. For example, a strong oscillation in the aircraft movement that was triggered by an unusual GCS command indicates an attack (or an irresponsible pilot). Similarly, transients in GPS signals followed by subtle position movements could be telltales of a GPS spoofing attack. Figure 3 shows how our R2U2 framework monitors the various inputs going into the UAS system (GCS and GPS), as well as sensor/actuator signals and status of the flight software for post-attack analysis. We next consider modeling for attacks and post-attack behavior, loosely following [14].

[3] We do not model attack scenarios via compromised flight software.

3.1 Attack Monitoring

As all attacks are initiated through the GCS or GPS inputs, we monitor the following attack surfaces. Because of zero-day attack mechanisms, this list will always be incomplete [4]. Note that the occurrence of such a situation does not mean that an actual attack is happening; other reasons like unusual flight conditions, transmission errors, or faulty hard- or software might be the reason.

Ill-formatted and illegal commands should not be processed by the FSW. Such commands could result from transmission errors or might be part of an attack. If such commands are received repeatedly a denial-of-service attack might be happening.

Dangerous commands are properly formatted but might cause severe problems or even a crash depending on the UAS mode. For example, a "reset-FSW" command sent to the UAS while in the air, will, most certainly, lead to a crash of the UAS because all communication and system parameters are lost. Thus, in all likelihood, this command indicates a malicious attack. Other dangerous commands are, for example, the setting of a gain in the control loops during flight. However, there are situations where such a command is perfectly legal and necessary.

Nonsensical or repeated navigation commands could point to a malicious attack. Although new navigation waypoints can be sent to a UAS during flight to update its mission, repeated sending of waypoints with identical coordinates, or weird/erroneous coordinates might indicate an attack.

Transients in GPS signals might be signs of GPS spoofing or jamming. Because the quality of UAS navigation strongly depends on the quality of the received GPS signals, sudden transients in the number of available satellites, or signal strength and noise ratios (Jamming-to-Noise Sensing [9]) might indicate a GPS-based attack.

It should be noted that these patterns do not provide enough evidence to reliably identify an attack. Only in correspondence with a matching post-attack behavior are we able to separate malicious attacks from unusual, but legal command sequences. We therefore also monitor UAS behavior.

3.2 System Behavior Monitoring

Our R2U2 models for monitoring post-attack behavior obtain their information from the UAS sensors, actuators, and the flight computer. In our current setting, we do not monitor those electrical signals directly, but obtain their values from the FSW. This simplification, however, prevents our current implementation from detecting a crash of the flight software initiated by a malicious attack. With our R2U2 framework we are able to monitor the following UAS behaviors, which might (or might not be) the result of a malicious attack.

Oscillations of the aircraft around any of its axes hampers the aircraft's performance and can lead to disintegration of the plane and a subsequent crash. Pilot-induced oscillations (PIO) in commercial aircraft have caused severe accidents and loss of life. In a UAS such oscillations can be caused by issuing appropriate command sequences or by setting gains of the control loops to bad values. Oscillations of higher frequencies can cause damage due to vibration or can render on-board cameras inoperative.

Deviation from flight path: In the nominal case, a UAS flies from one waypoint to the next via a direct path. Sudden deviations from such a straight path could indicate some unplanned or possibly unwelcome maneuver. The same observation holds for sudden climbs or descents of the UAS.

Sensor Readings: Sudden changes of sensor readings or consistent drift in the same direction might also be part of a post-attack behavior. Here again, such behavior might have been caused by, for example, a failing sensor.

Unusual software behavior like memory leaks, increased number of real-time failures, or illegal numerical values can possibly point to an on-going malicious attack. In the case of software, such a behavior might be a post-attack behavior or the manifestation of the attack mechanism itself. Therefore, security models involving software health are the most complex ones.

3.3 R2U2 Models

We capture the specific patterns for each of the attack and behavior observers with temporal logic and Bayesian networks. We also use these mechanisms to specify the temporal, causal, and probabilistic relationships between them. As a high-level explanation, an attack is detected if a behavioral pattern B is observed some time after a triggering attack A has been monitored. Temporal constraints ensure that these events are actually correlated. So, for example, we can express that an oscillation of the UAS (*osc* = *true*) occurs between 100-200 time steps after the control loop parameters have been altered (*param_change* = *true*). Any trace satisfying the following formula could indicate an attack.

$$\square(\textit{param\_change} \wedge \Diamond_{[100,200]} \textit{osc})$$

3.4 Modeling Variants and Patterns

The combination of signal processing, filtering, past-time and future-time MTL, and Bayesian reasoning provides a highly expressive medium for formulating security properties. Further generality can be achieved by grouping related indicators. For example, we can define groups of dangerous commands, unusual repeated commands, or events:

$$\begin{aligned}
\mathbf{dangerous_cmds} &= cmd_reset \vee cmd_calibrate_sensor \vee cmd_disarm \ldots \\
\mathbf{unusual_cmds_airborne} &= cmd_get_params \vee set_params \vee get_waypoints \ldots \\
\mathbf{unusual_cmds_periodic} &= cmd_nav_to \vee cmd_mode_change \vee invalid_packet_rcvd
\end{aligned}$$

This enables us to directly use these preprocessed groups in temporal formulas and feed them into a BN, thereby supporting simple reuse of common patterns and assist to create more comprehensive security models. The following example demonstrates how we use such patterns to specify that there shall be no dangerous commands between takeoff and landing.

$$\square[(CMD == \text{takeoff}) \rightarrow ((\neg\, \mathbf{dangerous\_cmds})\, \mathcal{U}\ landing\_complete)]$$

3.5 Bayesian Networks for Security Diagnosis

Most models of attack monitoring and post-attack behavior are capable of indicating that there might have been an attack, but cannot reliably detect one as such, because the observed patterns could have been caused by a sensor failure, for example. However, we can use the Bayesian network (BN) reasoning engine of R2U2 to perform better probabilistic diagnosis. For details of Bayesian R2U2 models see [29]. The results of all the temporal observers are provided as inputs to the observable nodes (shaded in Figure 4) of the BN. The internal structure of the BN then determines how likely a specific attack or failure scenario is. Prior information on how likely a certain monitor fires helps to disambiguate the diagnosis. For example, a sudden change in measured altitude could be attributed to a failing barometric altimeter, a failing laser altimeter, a failing GPS receiver, or a GPS spoofing attack. In order to determine the most likely root cause, additional information about recently received commands, or the signal strength of the GPS receiver can be used. So, transients in GPS signal strength with otherwise healthy sensors (i.e., measured barometric and laser altitude coincide) make an attack more likely. On the other hand, strongly diverging readings from the on-board altitude sensors make a sensor failure more likely. With prior information added to the BN, we can, for example, express that the laser altimeter is much more likely to fail than the barometric altimeter or the GPS sensor. Also, GPS transients might be more likely in areas with an overall low signal strength. Since a BN is capable of expressing, in a statistically correct way, the interrelationships of a multitude of different signals and outputs of temporal observers, R2U2 can provide best-possible attack diagnosis results as sketched in Figure 4.

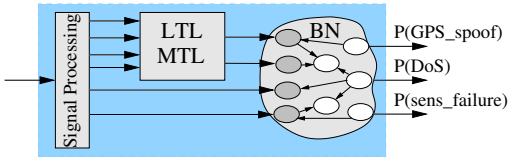

Fig. 4. Threat detection with R2U2 model

4 Experiments and Results

Our experiments can be run either in a software-in-the-loop (SITL) simulation or directly on the UAS; most of the experiments in this paper were executed on our Ironbird processor-in-the-loop setup, which consists of the original UAS flight computer hardware components in a laboratory environment. In all configurations, the produced data traces were forwarded via a UART transmission to the R2U2 framework running on an Adapteva Parallella Board [1]. R2U2 is implemented on this credit-card sized, low-cost platform where the actual monitoring is performed inside the Xilinx[4] zynq xc7z010 FPGA. Our R2U2 implementation (Figure 2) uses 40% of the FPGA's slice registers and 64% of its slice look-up tables (LUTs). These numbers are independent of the size and structure of the LTL and MTL formulas. The implementation in this paper used

[4] http://www.xilinx.com

128 input signals through the UART to the FPGA, though this number could be extended for other implementations. The R2U2 framework is running with a maximum frequency 85.164MHz. A Ubuntu Linux installation on the Parallella board is used for the interface configuration, signal preprocessing, and evaluation of arithmetic circuits.

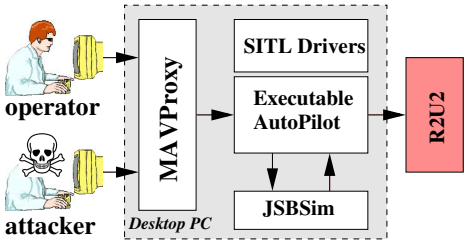

Fig. 5. R2U2 SITL test setup

In our SITL simulation (Figure 5), the UAS flight behavior is simulated by the open source JSBSim [11] flight dynamics model. All hardware components are emulated by SITL low-level drivers, which enables us to inject the desired behavior without the risk of damaging the aircraft during a real test flight. The operator's GCS is connected to the UAS via an open source MAVLink proxy [17]. We also connect a second GCS to the proxy in order to simulate the attackers injected MAVLink packets.

4.1 Dangerous MAV Commands

In addition to commands controlling the actual flight, the MAVLink protocol allows the user to remotely setup and configure the aircraft. In particular, parameters that control the feedback loops inside the FSW can be defined, as they need to be carefully adjusted to match the flight dynamics of the given aircraft. Such commands, which substantially alter the behavior of the UAS can, when given during flight, cause dangerous behavior of the UAS or a potential crash. In 2000, a pilot of a Predator UAS inadvertently sent a command "Program AV EEPROM" while the UAS was in the air. This caused all FSW parameters and information about communication frequencies to be erased on the UAS. Communication to the UAS could not be reestablished and the UAS crashed causing a total loss $3.7M [32]. If parameters for the FSW control loops are set to extreme values during flight, the aircraft can experience oscillations that could lead to disintegration of the UAS and subsequent crash. Therefore, such commands might be welcome targets for a malicious attack.

In this experiment, we set up our R2U2 to capture and report such dangerous behavior. Our security model consists of two parts: (a) detection that a potentially dangerous MAV command has been issued, and (b) that a dangerous behavior (in our case, oscillation around the pitch axis) occurs. Each of the parts seen individually does not give an indication of an attack: MAV commands to change parameters are perfectly legal in most circumstances. On the other hand, oscillations can be caused by turbulence, aircraft design, or the pilot (pilot-induced-oscillations). Only the right temporal combination of both pieces of information allows us to deduce that a malicious attack has occurred: after receiving the "set parameter" command, a substantial and persistent oscillation must follow soon thereafter. In our model we use the specification $\square(C \wedge \Diamond_{[0,1200]}(\square_{[0,300]} O))$ where O is the occurrence of oscillations and C the event of receiving a "set parameter" command. We require that the oscillation persists for at least 300 time steps and is separated by the command by not more than 1200 time steps.

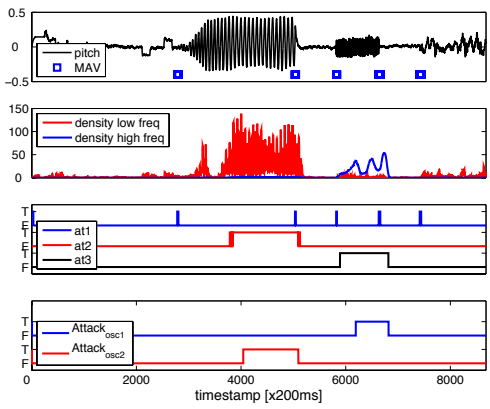

Fig. 6. UAS behavior after malicious setting of gain parameters

The event C can be directly extracted from the stream of received MAV commands; oscillations can be detected with the help of a Fast Fourier Transform (FFT) on the pitch, roll, or yaw angular values. Figure 6 shows how such an attack occurs. The top panel shows the UAS pitch as well as the points in time when a "set-parameter" command has been received (blue boxes). Caused by a malicious command (setting pitch gain extremely high) issued at around $t = 2800$, a strong low-frequency up-down oscillation appears in the pitch axis. That excessive gain is turned off at around $t = 5100$ and the oscillation subsides. Shortly afterwards, at $t = 5900$, a malicious setting of a damping coefficient causes smaller oscillations but at a higher frequency. This oscillation ramps up much quicker and ends with resetting that parameter. In the second panel, two elements of the power spectrum obtained by an FFT transform of the pitch signals are shown. The signals, which have been subjected to a low-pass filter clearly indicate the occurrence of a low (red) and high (blue) frequency oscillation. The third panel shows the actual Boolean inputs for R2U2: "set-parameter received" C (green), "Low-frequency-oscillation" O_L (red), and "high-frequency-oscillation" O_H (blue). The bottom panel shows valuations of formulas $\square(C \wedge \Diamond_{[0,1200]}(\square_{[0,300]} O_L))$ and $\square(C \wedge \Diamond_{[0,500]}(\square_{[0,200]} O_H))$ as produced by the R2U2 monitor. On the latter property the maximal lead time of the malicious attack has been set to only 500 time steps to reduce the number of false alarms, because the high-frequency oscillation ramps up almost immediately. We estimate that 10 person-hours were spent writing, debugging, and revising the two temporal logic properties used for this experiment and approximately 30 hours were spent on experimental setup and simulation.

4.2 DoS hijack Attack

Attackers continuously find new ways to break into and compromise a system. Therefore, it is challenging to account for every possible attack scenario, since there can always be an unforeseen loophole. The following experiment shows how our R2U2 framework can detect an intrusion without the need of an explicit security model for each specific scenario. Here, we will look at possible indicators that can be grouped into patterns as described earlier.

In our simulation we initiate a sophisticated attack to hijack the UAS by first trying to establish a link from the attacker's GCS to the UAS. Because the attacker has to cope with issues like an incorrect channel, a different version of the protocol, or link encryption, a large number of bad command packets will be received within a short

time frame. The top panel of Figure 7 shows such a typical situation (black). The R2U2 security model could use, for example, the following formula for detection given the rate of received bad command packages R_b per time step:

$$F_1 \equiv \square_{[0,10]}(R_b = 0 \vee (R_b \geq 1 \, \mathcal{U}_{[0,10]} R_b = 0))$$

The formula F_1 means that no more than one bad command is received within each time interval of length 10 time steps.

Next, an attacker could try to gather information about the UAS, e.g., by requesting aircraft parameters or trying to download the waypoints using the MAVLink protocol. This activity is shown in Figure 7 as blue spikes between time step 1000 and 1300. For our model, we use the input command groups defined earlier. With C_u as the event of receiving an unusual command, we state that no unusual command should be received after takeoff until the UAS has landed.

$$F_2 \equiv \square((CMD == \text{takeoff}) \rightarrow \ (\neg \, C_u) \, \mathcal{U} \ \text{landing\_complete})$$

Finally, an attacker may flood the communication link in a way similar to a Denial of Service (DoS) attack by sending continuous requests C_{nav} to navigate to the attacker's coordinates, combined with requests $C_{homeloc}$ to set the home location of the UAS to the same coordinates. This phase of the attacks results in a continuously high number of navigation commands starting around $t = 1400$ as shown in the bottom panel of Figure 7. For attack detection, we specify formulas, either explicitly detecting an unusual period of navigational commands (F_3), or detecting a group of previously-defined unusual periodic commands (F_4). F_3 states that there shall be no continuous navigation requests for more than 30 time steps: $\square_{[0,30]} C_{nav}$. Finally, F_4 states that there shall be no continuous unusual periodic events for more than 60 time steps: $\square_{[0,60]} C_u$. The formulas F_1, F_2, F_3, and F_4 are not reliable indicators of an ongoing attack if viewed individually. Only by considering the overall pattern, can we calculate a high probability for an ongoing attack. In order to accomplish this, we feed the results of these formulas into a Bayesian network for probabilistic reasoning. We estimate that 6 person-hours were spent writing, debugging, and revising the four temporal logic properties used for this experiment and approximately 15 hours were spent on experimental setup and simulation.

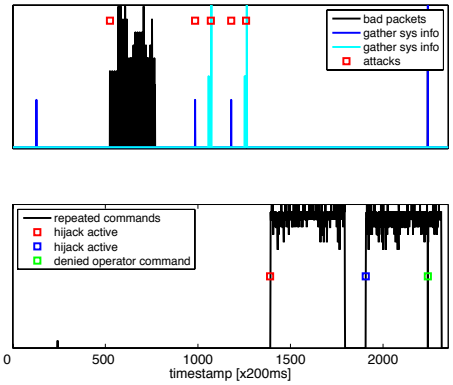

Fig. 7. UAS DoS hijack results

Even if the attack was detected by the UAS operator, all attempts to change the UAS to its original course would immediately be overwritten by the attacker's high-rate navigation commands. To due the altered home coordinates, any attempt of the UAS to return to the launch site would fail as well. Rather it would fly to the target location desired by the attacker. Furthermore, the simulation of this scenario showed that

besides crashing the UAS intentionally, there was no simple way for the UAS operator to prevent this kind of hijacking. In particular, for autonomous missions, where the UAS is flying outside the operator's communication range, it is essential that the UAS is capable of detecting such an attack autonomously.

In order to protect a UAS from attacks against command link jamming, a UAS is sometimes deliberately put into a complete autonomous mode, where it does not accept any further external commands [9]. Ironically, what was intended to be a security measure could inhibit the operator's attempts to recover a UAS during such an attack. However, R2U2 enables the UAS to detect an ongoing attack autonomously in order to enable adequate countermeasures.

4.3 GPS Spoofing

GPS plays a central role in the control of autonomous UAS. Typically, a flight plan for a UAS is defined as a list of waypoints, each giving a target specified by its longitude, latitude, and altitude. The FSW in the UAS then calculates a trajectory to reach the next waypoint in sequence. In order to accomplish this, the UAS needs to know its own position, which it obtains with the help of a GPS receiver. Due to limited accuracy, only GPS longitude and latitude are used for navigation; the UAS's current altitude is obtained using the barometric altimeter.

For the control of UAS attitude, the UAS is equipped with inertial sensors. Accelerometers measure current acceleration along each of the aircraft axes; gyros measure the angular velocity for each axis. Integration of these sensor values yields relative positions and velocities. These data streams are produced at a very fast rate and are independent from the outside interference but very noisy. Thus, the inertial sensors alone cannot be used for waypoint navigation. Therefore, the FSW uses an Extended Kalman Filter (EKF) to mix the inertial signals with the GPS position measurements. If the inertial measurements deviate too much from the GPS position, the filter is reset to the current GPS coordinates.

Several methods for attacking the GPS-based navigation of a UAS are known, including GPS jamming and GPS spoofing. In a jamming attack, the signals sent from the GPS satellites are drowned out by a powerful RF transmitter sending white noise. The UAS then cannot receive any useful GPS signals anymore and its navigation must rely on compass and dead reckoning. Such an attack can cause a UAS to miss its target or to crash. A more sophisticated attack involves GPS spoofing. In such a scenario, an attacker gradually overpowers actual GPS signals with counterfeit signals that have been altered to cause the UAS to incorrectly estimate its current position. That way, the UAS can be directed into a different flight path.

This type of attack became widely known when Iran allegedly used GPS spoofing to hijack a CIA drone and forced it to land on an Iranian airfield rather than its base [6,31]. Subsequently, researchers from the University of Texas at Austin successfully demonstrated how a $80M yacht at sea,[5] as well as a small UAS can be directed to follow a certain pattern due to GPS spoofing [13]. Because civil GPS sig-

[5] http://www.ae.utexas.edu/news/features/humphreys-research-group

nals are not encrypted it is always possible to launch a GPS spoofing attack. For such an attack, only a computer and a commercially available GPS transmitter is necessary. We can employ our R2U2 framework to detect realistic GPS spoofing attacks like the common attack scenarios described in [13]; whereas that paper discusses attack detection in theory we demonstrate it via hardware-in-the-loop simulation on-board our IronBird UAS. Here we focus on attack detection; techniques to avoid or mitigate GPS spoofing are beyond the scope of this paper.

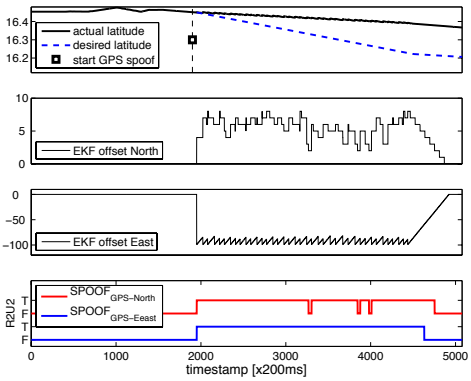

Fig. 8. Set of traces that indicate GPS spoofing

Our developed R2U2 model monitors the quality of the GPS signal and the inertial navigation information. For our experimental evaluation, we defined a UAS mission that flies, at a fixed altitude, toward the next waypoint south-south-west of the current UAS location. When spoofing occurs, the attacker modifies the GPS signal in such a way that it tricks the UAS into believing it is still flying a direct route as expected. In reality, however, the UAS is actually veering off to reach a target point defined by the attacker. Figure 8 shows the relevant signals during this mission. Here, we focus on the latitude as observed by the UAS. The top panel shows the point of the spoofing attack and the trace for the temporal development of the UAS latitude as observed by the UAS (black) and the actual UAS position (blue). A severe and increasing discrepancy can be observed as the effect of the attack. As the actual position (ground truth) is not available to the onboard FSW, R2U2 reasons about relationships with alternate signals that convey similar information. The inertial navigation unit produces an error or offset signal that reflects the deviation between the current position observed by GPS and the inertial sensors. The next two panels of Figure 8 shows that these offset signals can become substantially large during the actual spoofing period, when the GPS locations are gradually moved to the attacker's target. The bottom panel shows the spoofing detection output stream from R2U2. We estimate that 10 person-hours were spent on model development and approximately 45 hours were spent on experimental setup and simulation.

Again, these signals are not individually absolute indicators that an attack has happened. Flying in areas with weak GPS coverage, for example, in a mountainous or urban environment, could produce similar signals. Therefore, in our R2U2 models, we aim to take into account other observation patterns and use a Bayesian network for probabilistic reasoning. Information supporting the hypothesis of an attack could include prior loss of satellite locks, transients in GPS signals, or other types of attacks. In the case of the captured CIA drone, an Iranian engineer claimed to have jammed the drone's communications link in order to force the drone into an autopilot mode and then initiated the GPS spoofing attack [31].

5 Conclusion

We have extended our REALIZABLE, RESPONSIVE, UNOBTRUSIVE Unit (R2U2) to enable real-time monitoring and diagnosis of security threats. This includes the ability to reason about complex and subtle threats utilizing indicators from both the UAS system and its software. Our embedded implementation on-board a standard, flight-certifiable FPGA meets stated FAA requirements and efficiently recognizes both individual attack indicators and attack patterns, adding a new level of security checks not available in any previous work. Case studies on-board a real NASA DragonEye UAS provide a promising proof-of-concept of this new architecture.

The myriad directions now open for future work include considering software instrumentation to enable more FSW-related compromises and doing hardware-in-the-loop simulation experiments to detect these. We plan to extend this technology to other, more complex UAS and beyond, to other types of aircraft and spacecraft with different configurations and capabilities. A major bottleneck of the current R2U2 is the manual labor required to develop and test every temporal logic formula and BN; we are currently considering methods for making this a semi-automated process to better enable future extensions.

References

1. Adapteva: The Parallella Board, https://www.parallella.org/board
2. Ahmed, A., Lisitsa, A., Dixon, C.: TeStID: A High Performance Temporal Intrusion Detection System. In: Proc. ICIMP 2013, pp. 20–26 (2013)
3. APM:Plane, Open Source Fixed-Wing Aircraft UAV, http://plane.ardupilot.com
4. Bilge, L., Dumitras, T.: Before we knew it: An empirical study of zero-day attacks in the real world. In: Proc. CCS '12. pp. 833–844. (2012)
5. Bushnell, D., Denney, E., Enomoto, F., Pai, G., Schumann, J.: Preliminary Recommendations for the Collection, Storage, and Analysis of UAS Safety Data. Technical Report NASA/TM-2013-216624, NASA Ames Research Center (2013)
6. Eulich, W.: Did Iran just down a US drone by 'spoofing'? Christian Science Monitor (2012), http://www.csmonitor.com/World/Security-Watch/terrorism-security/2012/1204/Did-Iran-just-down-a-US-drone-by-spoofing-video
7. GAO: Air Traffic Control: FAA Needs a More Comprehensive Approach to Address Cybersecurity As Agency Transitions to NextGen. Tech. Rep. GAO-15-370, United States Government Accountability Office (2015), http://www.gao.gov/assets/670/669627.pdf
8. Geist, J., Rozier, K.Y., Schumann, J.: Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In: Proc. RV2014, pp. 215–230 (2014)
9. Humphreys, T.: Statement on the Vulnerability of Civil Unmanned Aerial Vehicles and Other Systems to Civil GPS Spoofing. University of Texas at Austin (2012)
10. Javaid, A.Y., Sun, W., Devabhaktuni, V.K., Alam, M.: Cyber Security Threat Analysis and Modeling of an Unmanned Aerial Vehicle System. In: Proc. HST2012, pp. 585–590., IEEE (2012)
11. JSBSim: Open Source Flight Dynamics Model, http://jsbsim.sourceforge.net
12. Karimi, N.: Iran Drone Capture Claim: State TV Airs Images Allegedly Extracted From U.S. Aircraft (video). The World Post (2013), http://www.huffingtonpost.com/2013/02/07/iran-drone-capture-claim_n_2636745.html

13. Kerns, A.J., Shepard, D.P., Bhatti, J.A., Humphreys, T.E.: Unmanned Aircraft Capture and Control Via GPS Spoofing. Journal of Field Robotics 31(4), pp. 617–636 (2014)
14. Kim, A., Wampler, B., Goppert, J., Hwang, I., Aldridge, H.: Cyber attack vulnerabilities analysis for unmanned aerial vehicles. Infotech@Aerospace (2012)
15. Lu, H., Forin, A.: The Design and Implementation of P2V, An Architecture for Zero-Overhead Online Verification of Software Programs. MSR-TR-2007-99, Microsoft Research (2007), http://research.microsoft.com/apps/pubs/default.aspx?id=70470
16. MAVLink: Micro Air Vehicle Protocol, https://github.com/mavlink
17. MAVProxy: A UAV Ground Station Software Package for MAVLink Based Systems, http://tridge.github.io/MAVProxy
18. Mengshoel, O.J., Chavira, M., Cascio, K., Poll, S., Darwiche, A., Uckun, S.: Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study. IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans 40(5), pp. 874–885 (2010)
19. Meredith, P.O., Jin, D., Griffith, D., Chen, F., Roşu, G.: An Overview of the MOP Runtime Verification Framework. International Journal on Software Tools for Technology Transfer 14(3), pp. 249–289 (2012)
20. Naldurg, P., Sen, K., Thati, P.: A temporal logic based framework for intrusion detection. In: FORTE, LNCS, vol. 3235, pp. 359–376. Springer (2004)
21. Olivain, J., Goubault-Larrecq, J.: The orchids intrusion detection tool. In: Proc. CAV, LNCS, vol. 3576, pp. 286–290. Springer (2005)
22. Pearl, J.: A constraint propagation approach to probabilistic reasoning. In: Proc. UAI. pp. 31–42. AUAI Press (1985)
23. Pellizzoni, R., Meredith, P., Caccamo, M., Rosu, G.: Hardware Runtime Monitoring for Dependable COTS-Based Real-Time Embedded Systems. RTSS, pp. 481–491 (2008)
24. Perry, S.: Subcommittee Hearing: Unmanned Aerial System Threats: Exploring Security Implications and Mitigation Technologies. Committee on Homeland Security (2015),http://homeland.house.gov/hearing/subcommittee-hearing-unmanned-aerial-system-threats-exploring-security-implications-and
25. Prince, B.: Air traffic control systems vulnerabilities could make for unfriendly skies [black hat]. Security Week (2012), http://www.securityweek.com/air-traffic-control-systems-vulnerabilities-could-make-unfriendly-skies-black-hat
26. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems. In: Proc. TACAS2014, ETAPS2014, (2014), LNCS, vol. 8413, pp. 357–372. Springer (2014)
27. Schumann, J., Mbaya, T., Mengshoel, O.J., Pipatsrisawat, K., Srivastava, A., Choi, A., Darwiche, A.: Software Health Management with Bayesian Networks. Innovations in Systems and Software Engineering 9(2), pp. 1–22 (2013)
28. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. In: Proc. PHM2013, pp. 381–401 (2013)
29. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. Int. J. of Prognostics and Health Management, 6(1) (2015)
30. Shachtman, N., Axe, D.: Most U.S. Drones Openly Broadcast Secret Video Feeds. Wired (2012), http://www.wired.com/2012/10/hack-proof-drone/
31. Shepard, D.P., Bhatti, J.A., Humphreys, T.E.: Drone Hack. GPS World 23(8), pp. 30–33 (2012)
32. USAF: Aircraft Accident Investigation: Rq-1l, s/n 96-3023. AIB Class A Aerospace Mishaps (2000), http://usaf.aib.law.af.mil/ExecSum2000/RQ-1L_Nellis_14Sep00.pdf