# Beyond Valid Commands: Security Limits of Command Mediation in ROS2-PX4 UAVs

Alberto Espinoza, Behzad Ousat, Raju Rangaswami, and Amin Kharraz
Florida International University
{aespi237, bousat, raju, mkharraz}@fiu.edu

***Abstract*—As unmanned aerial vehicles (UAVs) become increasingly prevalent across commercial, industrial, and public-sector applications, growing autonomy is pushing more mission-critical decision making onto companion computers and ROS2-based software stacks. We study the security limits of policy-based command mediation at the ROS2–PX4 boundary. Because a companion-computer compromise yields an already-authorized publisher, we consider a ROS2 controller that retains legitimate credentials and issues syntactically valid commands. Through architectural analysis and PX4 software-in-the-loop experiments, we characterize four attack families: direct policy violations, enforcement-path and observability gaps, command-semantic inconsistencies, and temporal-semantic evasions. Our results show that local policies effectively block explicitly represented violations, but their guarantees depend on mandatory mediation and correct interpretation of command semantics. More fundamentally, sequences of individually admissible commands can produce unsafe cumulative drift, and a digital twin fed the same admitted commands will track that attacker-chosen path rather than the authorized mission. These findings show that securing autonomous command paths requires not only local validation, but also temporal reasoning anchored to an attacker-independent mission reference. We discuss a practical architecture for a trusted mission representation that encodes authenticated constraints rather than only a fixed trajectory and remains outside the compromised planner's control.**

***Index Terms*—UAV security, ROS2, PX4, command mediation, digital twin**

## I. Introduction

Unmanned aerial vehicles (UAVs) are becoming increasingly common across commercial, industrial, and public-sector applications [1], and growing autonomy is a major driver of this expansion. Capabilities such as real-time perception, obstacle avoidance, adaptive mission planning, and coordinated navigation require substantially more computation than a conventional flight controller can provide. This has led to an emerging architecture in which a safety-critical flight controller handles stabilization and low-level control, while a more capable companion computer runs high-level autonomy software. PX4 [2] and ROS2 [3] provide a widely used open-source realization of this model: PX4 executes time-sensitive flight-control functions on embedded hardware, while ROS2 nodes running on Linux perform perception, planning, and mission management. The architecture enables rapid development of increasingly intelligent UAV behavior, but it also creates a security-critical command boundary between the autonomy software and the flight controller.

Securing the ROS2–PX4 boundary is challenging because an adversary who compromises an authorized ROS2 node [4]–[6] need not exploit PX4, modify firmware, or inject malformed traffic. A compromised planner can instead issue syntactically valid trajectory or velocity setpoints through the same topics, message types, and privileges used during legitimate operation. This threat is architectural rather than anecdotal: once a node holds valid middleware credentials, its commands are indistinguishable at the transport layer from those of a legitimate planner. A natural defense is *policy-based command mediation* [7]–[10], in which autopilot-bound commands are intercepted and checked against predefined constraints before being forwarded to the flight controller. Such mechanisms can effectively reject explicit violations, such as excessive velocity, altitude, or displacement, but their decisions are predominantly local to individual commands. An adversary that understands these policies may therefore remain within each admissible boundary while constructing a sequence of commands whose cumulative effect is unsafe. This raises a broader question for autonomous UAV security: how far can conventional command mediation protect the vehicle when the adversary controls an authorized autonomy component and deliberately behaves within the limits enforced by the mediator?

We investigate this problem by examining both the guarantees of policy-based command mediation and the conditions under which its protections can be evaded. Specifically, we ask *(RQ1)* what security guarantees can command-path mediation provide against a compromised but authorized ROS2 controller, and where do complete mediation or local policy coverage break down? We further ask *(RQ2)* under what command sequences, system assumptions, and runtime conditions can an adversary remain policy-compliant while nevertheless driving the UAV away from its intended behavior? We address these questions through architectural analysis of the ROS2–PX4 command path, analytical reasoning about adversarial command sequences, implementation-level examination of admission semantics, and controlled PX4 software-in-the-loop experiments.

Our analysis reveals three limitations of conventional policy-based command mediation. A mediator is only as strong as middleware access control: if an authorized node can publish directly to PX4-facing topics, the enforcement boundary collapses. Per-command policies also miss cumulative drift from individually admissible commands, and a

command-driven digital twin (a lightweight runtime model that predicts the next vehicle state from admitted commands and compares it with PX4 telemetry) does not close that gap, because its residual (the distance between predicted and reported state) measures tracking mismatch rather than mission intent. Securing the ROS2–PX4 boundary therefore requires mandatory mediation, temporal reasoning across command sequences, and an attacker-independent mission reference. We discuss a practical architecture for a trusted mission representation that can encode authenticated constraints and authorized replanning, provided that this reference cannot be rewritten by the compromised command generator.

## II. Background and Threat Model

### A. The ROS2–PX4 Command Path

Modern autonomous UAVs commonly separate flight-critical control from high-level autonomy. Timing-sensitive functions such as stabilization, state estimation, and actuator control execute on an embedded flight controller under an RTOS such as NuttX [11], while perception, planning, and learning-based components run on a Linux companion computer as ROS2 nodes communicating through DDS-backed topics [2], [3]. PX4 and ROS2 provide a representative realization of this architecture. Their communication is commonly supported through PX4's Micro XRCE-DDS bridge, which exposes selected uORB messages as DDS topics: vehicle state and telemetry flow toward ROS2, while commands such as `TrajectorySetpoint` flow toward PX4 [12], [13]. Figure 1 illustrates this command path together with the trusted computing base and threat surface described below.

This interface transfers control authority across a software boundary. Once a ROS2 setpoint reaches PX4, the flight controller processes it according to its current mode, controller constraints, geofences, and configured failsafes. The setpoint itself, however, does not encode why the command was issued or whether it remains consistent with the intended mission. A command may therefore be syntactically valid and physically executable while still being semantically unauthorized. In particular, a compromised planner can generate a sequence of individually reasonable setpoints whose cumulative effect moves the aircraft away from its intended operating region. At the PX4 interface, these commands are difficult to distinguish from those produced by an uncompromised planner because both use the same message types and command path.

A natural defense is *policy-based command mediation* [7]–[10]: candidate commands are intercepted before reaching PX4, evaluated against constraints such as altitude, velocity, or displacement, and forwarded only when compliant. Such mediation can reject explicit violations, but policies defined over individual commands or short transitions do not necessarily constrain their cumulative effect. An adversary may therefore remain within every local policy while producing unsafe behavior over time. This *semantic evasion* motivates our examination of where command mediation provides meaningful protection and where additional stateful and behavior-aware reasoning is required.

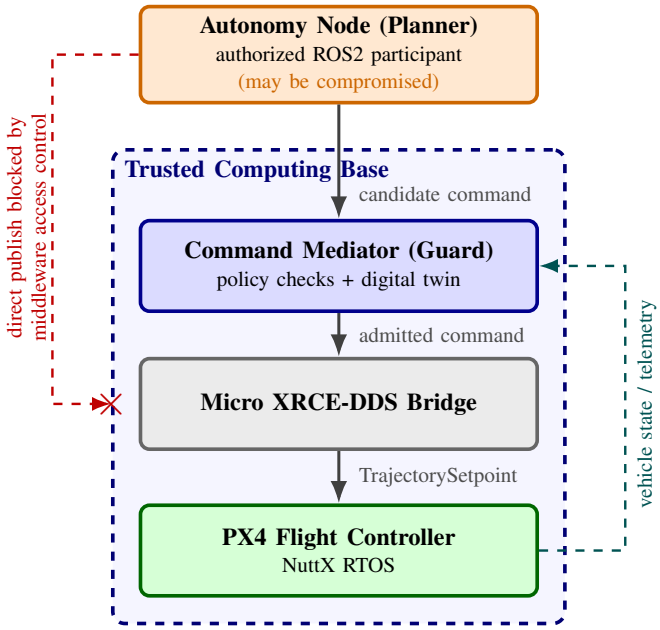



Fig. 1. The ROS2–PX4 command path. An authorized but potentially compromised autonomy node (planner) issues candidate commands to the command mediator (guard), which evaluates them against deterministic policies and a command-driven digital twin before forwarding admitted commands through the Micro XRCE-DDS bridge to PX4 (controller). The trusted computing base (dashed boundary) comprises the guard, the bridge, and PX4; correctly configured middleware access control prevents the autonomy node from publishing directly to the protected PX4-facing topic.

### B. Threat Model

**Adversary.** We consider an adversary that compromises an authorized ROS2 autonomy component, such as a planner or one of its software dependencies. The compromised component retains legitimate middleware credentials and can publish syntactically valid commands with attacker-selected values, timing, and rate. It may observe topics available to the compromised node and knows the guard's enforcement parameters; we therefore assume no security through obscurity. The adversary may coordinate admissible commands over time or generate high-rate traffic, but cannot modify PX4 firmware, compromise the flight-controller hardware, alter physical sensor measurements, or modify the runtime guard. We additionally exclude unrestricted root compromise of the companion computer, which would permit direct modification or bypass of any software-only mediator.

**Trusted-computing-base assumption.** Our threat model distinguishes compromise of the autonomy application from compromise of the companion-computer security boundary. The planner and its dependencies are untrusted and may be fully controlled by the adversary, including their ROS2 credentials and command-generation logic. The command guard, its policy state, and the middleware authorization configuration are trusted components protected by privilege separation and ROS2/DDS access control. In particular, the compromised planner is not granted the privileges required to modify the guard process, its configuration, or the publication permissions

that enforce the guarded path. We therefore model the guard, the bridge, and PX4 as the trusted computing base, as illustrated in Figure 1. If an adversary obtains unrestricted root or equivalent host-level privileges on the companion computer, this assumption no longer holds; such a compromise is outside our threat model.

This distinction is essential to our security question: we ask what mediation can guarantee after compromise of an authorized control application, not after compromise of the entire host on which the mediator executes.

**Security Objective.** Our objective is to determine whether every autopilot-bound command is mediated and whether the mediator can detect not only explicit policy violations but also unsafe behavior that emerges from sequences of individually admissible commands. We consider an evasion successful when an authorized but compromised controller causes the vehicle to violate an independently specified operating constraint while its commands continue to pass the mediator without triggering an alert.

**Attack Families.** Rather than treating each attack as an isolated implementation defect, we group the evaluated scenarios according to the underlying security mechanism they exploit. This organization allows us to distinguish failures of explicit policy enforcement from deeper gaps in mediation, command semantics, and temporal reasoning.

**A. Direct policy violations.** These attacks deliberately exceed a constraint that the guard explicitly represents. They include the gross multi-field violation, the single large position spike, and finite-field velocity injection. We also use *fast drift*, a high-rate command sequence whose implied motion exceeds the twin residual threshold, as a positive-control case for the digital twin. Fast drift creates sufficient *command–state disagreement* (the gap between the commanded setpoint and the state the twin predicts from that command) to exceed the residual threshold. The core idea is to establish the guard's baseline capability: when the malicious property is directly encoded in an admission predicate or residual threshold, the corresponding violation should be detected.

**B. Enforcement-path and observability bypasses.** This family includes the direct-publish bypass and the transient-publisher race. Both exploit the fact that a security mechanism can reason only about events that traverse, or become visible to, its enforcement path. Direct publication reaches a PX4-facing interface without invoking the guard, whereas a short-lived publisher may appear and disappear between topology scans. The common mechanism is therefore incomplete mediation or observation, rather than malicious command content.

**C. Command-semantic and cross-mode evasions.** This family combines the effective-velocity and NaN-poisoned attacks. Both exploit a mismatch between the physical meaning of a command and the representation used by the admission policy. In the effective-velocity case, repeated position increments can imply a motion rate substantially above the nominal velocity limit because displacement and velocity are checked through different policy paths. In the NaN case, PX4's mode-dependent use of non-finite fields can cause ordinary ordered comparisons to fail open [14] when finiteness and command mode are not validated explicitly. The common issue is that numerically correct thresholds are insufficient when timing, mode, and field semantics are interpreted inconsistently.

**D. Temporal-semantic evasions.** This family combines the lateral-ratchet and rate-limited twin-drift attacks and captures the central gap studied in this paper. The lateral ratchet issues a sequence of individually admissible position increments whose cumulative effect eventually violates an independently specified operating boundary. The rate-limited drift targets the stateful monitor. The digital twin is reactive rather than predictive-blocking: it is driven by already-admitted commands and compares its predicted state with PX4-reported telemetry after the fact; it does not simulate a command offline and veto forwarding before the vehicle moves. Because that prediction is driven by the same admitted command stream as the vehicle, the residual can remain bounded even while cumulative mission deviation continues to grow. The common mechanism is a mismatch between local or command-relative consistency and mission-level consistency. Each individual command may appear admissible, and the vehicle may faithfully execute the resulting trajectory, while the overall behavior remains inconsistent with the authorized mission.

## III. Methodology

Our methodology is designed to characterize both the guarantees and the failure modes of command mediation at the ROS2–PX4 boundary. We combine architectural analysis, implementation-level inspection, analytical modeling, and controlled experiments in PX4 software-in-the-loop (SITL). This combination allows us to distinguish failures caused by deployment assumptions, policy semantics, and runtime detection mechanisms from failures that depend on a particular vehicle trajectory or experimental configuration.

### A. Research Questions

*a) RQ1: What security guarantees does command mediation provide against a compromised but authorized ROS2 controller, and where do they break down?:* We examine two properties of command mediation: *complete mediation*, which requires every PX4-bound command to traverse the guard, and *policy coverage*, which requires the enforced rules to capture the unsafe behavior of interest. We evaluate bypass conditions and policy behavior across command modes, rates, and sequences to determine where local enforcement ceases to provide a meaningful security guarantee.

*b) RQ2: When can locally admissible commands cause unsafe behavior without triggering stateful detection?:* We examine whether a command-driven digital twin can detect malicious behavior when its reference is derived from the same admitted command stream that drives the vehicle. We analyze and experimentally evaluate adversarial sequences that remain locally policy-compliant while producing cumulative mission deviation, identifying when temporal state alone is insufficient without an attacker-independent mission reference.

Together, the two questions separate *enforcement* from *detection*. RQ1 asks whether unsafe commands are forced through an effective policy decision, whereas RQ2 asks whether behavior that passes that decision can be recognized as unsafe over time. This distinction allows us to determine which guarantees arise from command mediation and which require additional behavioral or mission-level context.

### B. Experimental Setup

We evaluate the command mediator using ROS2 Jazzy [3] and PX4 SITL [2] with the Gazebo [15] gz_x500 multirotor model. The ROS2 components are implemented in Python using rclpy, and PX4 communicates with ROS2 through the eProsima Micro XRCE-DDS bridge [12], [13]. The controller publishes trajectory commands to `/uav/control_intent`, which are intercepted by the guard and, when permitted, relayed to the PX4-facing `/fmu/in/trajectory_setpoint` interface. PX4 returns vehicle position and status telemetry to support closed-loop monitoring and the digital-twin analysis. The guard runs as a standard ROS2 node with message-level visibility over the DDS bus, while `rosbag2` with MCAP storage independently records the experimental traffic for post-run analysis and validation.

### C. Data Collection and Measurements

For each experiment, we collect the complete command and vehicle-state trace from both sides of the mediation boundary. The trace records candidate commands received by the guard, permit-or-block decisions, commands forwarded toward PX4, PX4-reported position and velocity, digital-twin predictions and residuals, publisher observations, and relevant vehicle-status and heartbeat messages. We additionally record message timestamps and guard callback timing so that security outcomes can be correlated with runtime behavior. An independent observer subscribes to the protected PX4-facing command interface when appropriate so that successful command delivery is determined independently of the guard's own logs.

For RQ1, the principal measurements are *attack success rate* and *policy-bypass rate*. An attack succeeds at the enforcement layer when an adversary-controlled command reaches the protected PX4-facing interface without receiving the intended mediation decision. For attacks that traverse the guard, we record the fraction of explicitly unsafe commands rejected and the fraction of locally admissible sequences that eventually violate the independently specified operating constraint. We additionally measure block and forwarding latency to determine whether mediation introduces meaningful delay into the command path.

For RQ2, the primary measurements are the *digital-twin residual* $\|p_{\text{twin}} - p_{\text{PX4}}\|$ (the distance between the twin's predicted position and the PX4-reported position), detection latency, and *displacement before detection*. The latter captures how far the vehicle has deviated from the authorized operating region when an alert is raised and is therefore more informative than detection latency alone. We measure these quantities as functions of adversarial drift rate and vehicle response and compare them with benign trajectories to characterize the false-positive tradeoff. Successful semantic evasion occurs when the vehicle violates the independently defined mission or operating constraint while all commands remain admitted and the runtime detector produces no corresponding alert.

## IV. Analysis

We organize the analysis around the four attack families introduced in Section II-B. This grouping separates failures that arise because a configured policy is explicitly violated from failures caused by missing mediation, incomplete command semantics, or insufficient temporal context.

### A. RQ1: What Does Command Mediation Actually Guarantee?

**Direct Policy Violations.** For each encoded violation, the guard evaluates the corresponding constraint, records a block, and the independent egress observer confirms that the rejected command does not reach PX4. Fast drift likewise trips the twin when command–state disagreement exceeds the residual threshold for the required sustain interval. The subsequent evasions are therefore not caused by a non-functional guard. Command mediation thus provides a useful local guarantee: an explicitly encoded per-message violation can be rejected before it reaches PX4.

**Finding 1.** The guard reliably detects violations that are explicitly represented by its admission predicates or residual threshold. The remaining failures therefore arise at the boundary of what those predicates can observe or express, rather than from an inability to enforce known local constraints.

**Enforcement-Path and Observability Bypasses** The direct-publish experiment shows that a relay does not, by itself, establish complete mediation. When the compromised node retains permission to publish to a PX4-facing topic, the egress observer records the command while the guard records no permit-or-block decision: the policy is never invoked. Complete mediation therefore requires publication-side access control on every control-capable bridged topic, not only `TrajectorySetpoint`. Protecting one command interface while another remains directly writable simply relocates the bypass. For the transient-publisher race, if the monitor samples every $T_p$ seconds and a publisher remains active for $T < T_p$, then under a uniformly distributed arrival phase its probability of observation is approximately

$$P(\text{detect}) = \min\left(1, \frac{T}{T_p}\right).$$

Reducing $T_p$ narrows the observation window but does not make topology monitoring an enforcement mechanism. More importantly, the primary adversary in our threat model, an already-authorized and already-running ROS2 controller, creates no new topology event when it becomes malicious.

**Finding 2.** Command mediation is conditional on the surrounding middleware configuration. Complete mediation requires publication-side access control that makes the guarded

path unavoidable; topology monitoring can provide visibility into some participants, but cannot substitute for that property.

**Command-Semantic and Cross-Mode Evasions** A position-mode policy that accepts a displacement of at most $d_{\max}$ per update implicitly permits an effective speed of approximately

$$v_{\text{eff}} = d_{\max} f,$$

where $f$ is the command-forwarding rate. If velocity mode independently enforces a speed limit $s_{\max}$, the two policies are physically consistent only when

$$d_{\max} f \leq s_{\max}.$$

A sender that increases $f$ can therefore induce motion that exceeds the intended velocity constraint while every position-mode increment remains individually admissible. The weakness is not that either numerical bound is necessarily unreasonable in isolation; the policy fails to interpret the two command representations according to a common physical constraint.

The NaN-poisoned case is complementary. No choice of $d_{\max}$ or $s_{\max}$ repairs it: ordinary ordered comparisons fail open [14] because comparisons against NaN do not evaluate as threshold violations. Correct validation requires explicit finiteness checks and consistency between the active mode and the fields being interpreted.

**Finding 3.** Per-command enforcement is only as sound as its interpretation of the command representation. Cross-mode inconsistency and fail-open handling of non-finite fields allow policy-compliant messages to violate the physical property the policy was intended to enforce.

> **Answer to RQ1.** Command mediation provides a meaningful but bounded security guarantee. It reliably blocks violations that are explicitly represented by its local predicates, but only when middleware authorization makes the guarded path mandatory. Even under complete mediation, the guarantee is limited by the semantics encoded in the policy: publication timing, control mode, field representation, and cumulative behavior must be represented explicitly if they are part of the intended safety property. Thus, complete mediation and correct local validation are necessary, but they do not by themselves establish trajectory-level or mission-level safety.

### *B. RQ2: When Can Locally Admissible Behavior Evade Stateful Detection?*

RQ2 examines whether commands can remain locally admissible while their cumulative effect becomes unsafe. We focus on two cases: whether local transition policies constrain aggregate behavior, and whether a command-driven digital twin provides an independent signal of mission deviation.

**The Lateral Ratchet: Local Safety Does Not Bound Cumulative Behavior.** Consider a displacement policy that accepts command $\mathbf{c}_k$ when

$$\|\mathbf{c}_k - \mathbf{c}_{k-1}\| \leq d_{\max}.$$

An attacker can choose

$$\mathbf{c}_k = \mathbf{c}_0 + k\boldsymbol{\delta}, \qquad \|\boldsymbol{\delta}\| < d_{\max},$$

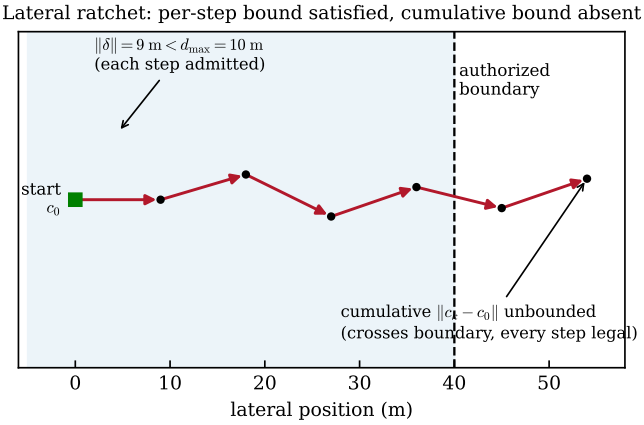



Fig. 2. Lateral-ratchet evasion. Every transition satisfies $\|\mathbf{c}_k - \mathbf{c}_{k-1}\| \leq d_{\max}$, yet cumulative displacement grows until the authorized boundary is crossed.

so every transition is admissible while

$$\|\mathbf{c}_k - \mathbf{c}_0\| = k\|\boldsymbol{\delta}\|$$

continues to grow.

Figure 2 illustrates the key limitation: each accepted setpoint becomes the reference for the next decision, allowing the admissible region to move with the attacker. Tightening $d_{\max}$ only reduces the step size; it does not bound the number of admissible steps. Preventing this behavior therefore requires an absolute or mission-relative constraint.

**Finding 4.** Per-step validity does not imply trajectory-level safety: a sequence of locally admissible commands can produce unbounded cumulative deviation.

**Rate-Limited Drift: The Twin Measures Tracking Consistency.** For a first-order twin and vehicle model,

$$\dot{\hat{p}} = \frac{c - \hat{p}}{\tau}, \qquad \dot{p} = \frac{c - p}{\tau_v}.$$

For an attacker-controlled ramp $c(t) = v_a t$, both systems asymptotically track the command with constant lag,

$$\hat{p}(t) \to v_a(t - \tau), \qquad p(t) \to v_a(t - \tau_v),$$

yielding

$$r_\infty = v_a|\tau_v - \tau|.$$

The residual therefore depends on drift rate and model mismatch rather than cumulative displacement. Under the stated model, any drift satisfying

$$v_a < \frac{R_{\text{thr}}}{|\tau_v - \tau|}$$

remains below the steady-state detection threshold even as mission deviation continues to grow. Fast drift can therefore be detected, while sufficiently slow drift is not: if the vehicle follows the admitted stream faithfully, low residual indicates tracking consistency, not mission authorization.

**Finding 5.** A rate-limited attack can remain below the twin residual threshold while cumulative mission deviation grows;

the twin therefore measures tracking consistency rather than mission legitimacy.

**Real-world disturbances and threshold calibration.** The SITL experiments provide a controlled setting in which the residual threshold can be interpreted without substantial environmental uncertainty. A physical deployment would introduce additional residual variation from wind, actuator dynamics, state-estimator noise, latency variation, and model mismatch. These effects would require calibration of the residual threshold against benign operating conditions rather than direct transfer of the SITL threshold. A lower threshold could improve sensitivity to slow deviations but increase false positives under normal disturbances, whereas a higher threshold would tolerate more benign variation at the cost of increasing the attacker's allowable drift rate. These practical effects change the detection tradeoff but do not remove the structural limitation identified here: when the twin is driven by the same attacker-controlled command stream as the vehicle, a small residual establishes tracking consistency, not mission legitimacy.

**Composed Temporal Evasion.** The two limitations compose naturally. The lateral ratchet keeps each command within the local admission region, while the twin follows the same attacker-selected sequence as the vehicle. Because the attacker is an already-authorized ROS2 controller, no new topology event is required. The resulting behavior can therefore remain valid at the message level, locally policy-compliant, and consistent with the runtime model while violating an independent operating constraint.

**Finding 6.** Policy compliance and model consistency can coexist with mission-level violation when an authorized attacker controls the command sequence used by both the mediator and its runtime reference.

**Answer to RQ2.** Stateful monitoring improves the guard's ability to reason across time, but temporal state alone is insufficient when it remains anchored to an attacker-influenced command stream. The lateral-ratchet attack shows that local transition constraints do not bound cumulative behavior, while the rate-limited drift result shows that a command-driven twin can track an attacker-chosen path rather than the authorized mission. Detecting this class of evasion requires temporal reasoning anchored to an attacker-independent mission reference, a trusted representation that the compromised command generator cannot redefine.

### C. Why Threshold Re-Tuning Is Insufficient

The four attack families show that these gaps are not simply threshold-selection problems; each exposes a different missing property or source of information. Direct policy violations are handled well because the unsafe property is already represented by the guard. Enforcement-path bypasses are architectural: no threshold can reject a command the mediator never sees. Command-semantic evasions require correct interpretation of timing, control mode, and field representation, while temporal-semantic evasions require cumulative state and an attacker-independent mission reference. These results suggest that the required defenses operate at different architectural layers. Complete mediation requires publication-side authorization that makes the guarded path unavoidable. Command-semantic safety requires validation that jointly reasons about rate, control mode, field representation, and finiteness. Trajectory-level safety requires state accumulated across commands rather than only transition-local checks. Finally, mission-level safety requires a reference that is not derived solely from the suspect command stream. A trusted mission representation can provide this reference while still permitting dynamic replanning, provided that the constraints governing authorized behavior remain outside the attacker's control. Thus, the central requirement is not a particular threshold or monitoring algorithm, but the availability of security-relevant information that the compromised command generator cannot redefine.

## V. Discussion

**Practical realization.** The attacker-independent mission does not need to be a second copy of the planner or a fixed list of waypoints. Its security property is that the reference used to evaluate behavior is established and protected independently of the potentially compromised command generator. A practical architecture can therefore separate mission authorization from mission execution: authorized mission intent is established once and stored as an authenticated mission representation, while the autonomy planner consumes that representation to generate commands. The guard or a trusted runtime monitor then evaluates the evolving vehicle state against the protected representation rather than treating the latest planner command as the reference for subsequent decisions. Concretely, the representation is consumed by two TCB components: the command guard, which admits or rejects candidate commands, and a temporal monitor, which evaluates the evolving vehicle state against the protected constraints. Admitted commands then proceed to PX4. The compromised planner may still generate candidate commands, but it cannot modify the representation against which those commands and the resulting trajectory are judged.

**Trust boundary.** The trusted representation could reside in a protected mission store on the companion computer, in a flight-controller-resident mission state, or in a separate hardware-isolated monitor, depending on the deployment's trust boundary. For systems in which mission metadata must cross software boundaries, an authenticated or cryptographically signed mission descriptor can prevent the compromised planner from silently replacing the reference used for enforcement. These mechanisms are implementation choices rather than requirements of our attack construction; the required property is that the reference remains outside the attacker's command-generation loop. Importantly, simply placing the reference in an ordinary file or ROS2 topic on the compromised host would not provide this property. If the compromised planner can modify both the commands and the reference against which those commands are evaluated, the monitor again becomes self-referential. The mission reference must therefore be protected by a stronger trust boundary than the planner.

**Dynamic missions.** An attacker-independent mission reference need not prescribe every future setpoint. For adaptive

missions, the trusted representation can instead encode mission invariants, authorized operating regions, temporal constraints, and permitted classes of replanning. For example, a mission may authorize adaptive waypoint selection within a geographic region and altitude envelope while prohibiting departure from that region or excessive cumulative displacement. The planner can therefore adapt its commands in response to observations while the trusted monitor independently evaluates whether the resulting state trajectory remains consistent with the authorized mission constraints. In this formulation, independence refers to the source of the authorization constraints, not to the absence of adaptation. This distinction explains why the lateral-ratchet attack is not merely a limitation of waypoint-based missions: the attack exploits the absence of an independently enforced cumulative constraint. Whether the authorized reference is a trajectory, region, invariant, or temporal specification, the monitor must retain some mission-level property that the attacker cannot shift incrementally with each accepted command.

## VI. Related Work

### A. ROS2 and Robotic Middleware Security

The security of the ROS2 middleware layer that our command mediator sits above has also been studied on its own. Deng et al. [4] identify four SROS2 vulnerabilities that bypass ROS2's DDS-based access-control and confidentiality guarantees. Sandoval and Thulasiraman [5], using a PX4/Gazebo testbed similar to ours, show that ROS2's authentication and access control successfully block rogue-node injection and message spoofing by *unauthorized* participants. Mayoral-Vilches et al. [6] provide the SROS2 tooling for deriving the graph-level policies that determine which nodes may publish to which topics. These results support the trusted-computing-base assumption in Section II-B that middleware permissions are correctly configured, and they delimit the mediation guarantee RQ1 examines: as our bypass experiments show, the guard's protection collapses without them. None, however, address our case. Their threat model is an unauthenticated *outsider*, which DDS access control is designed to stop, whereas ours already holds valid credentials, so its commands are indistinguishable at the transport layer from a legitimate planner's.

### B. UAV and Autopilot Security

Prior work has also targeted the MAVLink transport between a ground control station and the autopilot, and PX4's sensor-level defenses, including MAVLink authentication attacks and countermeasures [16], confidentiality schemes [17], formal protocol verification [18], GPS-spoofing of the state estimator [19], and ML-based intrusion detection [20]. These attacks target the authenticity of the command channel or the fidelity of sensing, both orthogonal to our adversary model, which forges neither and is already an authorized ROS2 participant. One result is nonetheless conceptually close to RQ2: Jung et al. [19] show that PX4's EKF-based spoofing detector can be evaded by keeping the injected error under its anomaly threshold, revealing the same structural weakness we find in a command-driven digital twin, namely that a model-based monitor is only as strong as its independence from the signal an adversary controls.

### C. Runtime Command Mediation and Cyber-Physical Monitoring

Most closely related to our work is prior research on runtime monitoring and enforcement for UAVs and cyber-physical systems. Vierhauser et al. [7] extend fleet-scale UAV monitoring to benign errors rather than adversarial injection at a single vehicle's command boundary. Yel and Bezzo [8] add fast runtime monitoring with replanning and recovery, but target faults and imminent constraint violations rather than an authorized controller that remains locally compliant. de Niz et al. [21] formalize per-command runtime-assurance *enforcers*, using the same geofence example as our per-command model, that substitute a safe fallback whenever a command would be unsafe. Such arguments stop where RQ1 locates the limit of per-command policies: they do not consider an adversary who stays inside the geofence at every step while executing a multi-step maneuver whose cumulative effect is unsafe, the sequences RQ2 constructs. Schumann et al. [10] and Yoon et al. [9] extend monitoring across time: R2U2 checks past- and future-time temporal-logic properties over sequences of events, while Yoon et al. predict whether future control inputs will violate a geofence over a finite horizon. Both are relevant to the cumulative behaviors RQ2 considers, but each assumes the monitored property is already specified rather than representing authorized mission intent independently of a compromised controller. Sieber et al. [22] use ontology-based rules to flag unsafe UAV behavior while keeping a human operator in the loop, but likewise do not consider a compromised yet authorized ROS2 controller or sequences of individually valid commands. In contrast, our work shows how such sequences evade local enforcement and require temporal reasoning tied to an attacker-independent mission reference.

## VII. Conclusion

Policy-based command mediation at the ROS2–PX4 boundary therefore provides a useful but bounded security guarantee. Complete mediation and correct local validation can prevent explicitly represented command violations, but they do not by themselves establish trajectory- or mission-level safety against an authorized controller. Our results show that cumulative drift can remain locally admissible and that a command-driven digital twin can remain consistent with an attacker-controlled trajectory while the vehicle departs from its authorized behavior. Addressing this gap requires temporal reasoning anchored to a protected representation of mission intent. Such a representation need not be a fixed trajectory: it can encode authenticated mission constraints, operating regions, invariants, and authorized replanning behavior, provided that the security reference cannot be rewritten by the compromised command generator. This establishes a practical architectural direction for extending command mediation from local command validity toward mission-level security.

## ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their feedback. This project was partially supported by Microsoft AI Security, the U.S. National Science Foundation (NSF) under Grant No. 2219920. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## REFERENCES

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[2] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6235–6240.

[3] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[4] G. Deng, G. Xu, Y. Zhou, T. Zhang, and Y. Liu, "On the (in)security of secure ROS2," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. Association for Computing Machinery, 2022, pp. 739–753.

[5] S. Sandoval and P. Thulasiraman, "Cyber security assessment of the robot operating system 2 for aerial networks," in *2019 IEEE International Systems Conference (SysCon)*. IEEE, 2019, pp. 1–8.

[6] V. Mayoral-Vilches, R. White, G. Caiazza, and M. Arguedas, "SROS2: Usable cyber security tools for ROS 2," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 253–11 259.

[7] M. Vierhauser, J. Cleland-Huang, S. Bayley, T. Krismayer, R. Rabiser, and P. Grünbacher, "Monitoring CPS at runtime: A case study in the UAV domain," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2018, pp. 73–80.

[8] E. Yel and N. Bezzo, "Fast run-time monitoring, replanning, and recovery for safe autonomous system operations," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1661–1667.

[9] H. Yoon, Y. Chou, X. Chen, E. W. Frew, and S. Sankaranarayanan, "Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for UAVs," in *Runtime Verification*, ser. Lecture Notes in Computer Science, vol. 11757. Springer, 2019, pp. 349–367.

[10] J. Schumann, P. Moosbrugger, and K. Y. Rozier, "Runtime analysis with R2U2: A tool exhibition report," in *Runtime Verification*, ser. Lecture Notes in Computer Science, vol. 10012. Springer, 2016, pp. 504–509.

[11] The Apache Software Foundation, "Apache NuttX: A real-time operating system (RTOS)," https://nuttx.apache.org/, accessed: 2026-08-12.

[12] PX4 Development Team, "uXRCE-DDS (PX4-ROS 2/DDS bridge)," https://docs.px4.io/main/en/middleware/uxrce_dds, accessed: 2026-08-12.

[13] Object Management Group (OMG), "DDS for extremely resource constrained environments (DDS-XRCE), version 1.0," https://www.omg.org/spec/DDS-XRCE/1.0, 2020, accessed: 2026-08-12.

[14] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[15] Open Source Robotics Foundation, "Gazebo: An open-source robotics simulator," https://gazebosim.org/, accessed: 2026-08-12.

[16] F. Du, J. Ge, W. Wang, Y. Zou, S.-Y. Chang, and W. Fan, "Exploiting the vulnerabilities in MAVLink protocol for UAV hijacking," in *2024 17th International Conference on Security of Information and Networks (SIN)*. IEEE, 2024, pp. 1–8.

[17] A. Allouch, O. Cheikhrouhou, A. Koubâa, M. Khalgui, and T. Abbes, "MAVSec: Securing the MAVLink protocol for Ardupilot/PX4 unmanned aerial systems," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 621–628.

[18] T. Wray and Y. Wang, "PAVE-MAVLink: Formal verification of MAVLink 2 for secure UAV communications," in *MILCOM 2025 – 2025 IEEE Military Communications Conference (MILCOM)*. IEEE, 2025, pp. 1389–1395.

[19] J. H. Jung, M. Y. Hong, H. Choi, and J. W. Yoon, "An analysis of GPS spoofing attack and efficient approach to spoofing detection in PX4," *IEEE Access*, vol. 12, pp. 46 668–46 677, 2024.

[20] J. Whelan, A. Almehmadi, and K. El-Khatib, "Artificial intelligence for intrusion detection systems in unmanned aerial vehicles," *Computers & Electrical Engineering*, vol. 99, p. 107784, 2022.

[21] D. de Niz, B. Andersson, and G. A. Moreno, "Safety enforcement for the verification of autonomous systems," in *Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything*, ser. Proceedings of SPIE, M. C. Dudzik and J. C. Ricklin, Eds., vol. 10643. SPIE, 2018, p. 1064303.

[22] C. Sieber, L. M. Vieira da Silva, K. Grünhagen, and A. Fay, "Rule-based verification of autonomous unmanned aerial vehicles," *Drones*, vol. 8, no. 1, p. 26, 2024.