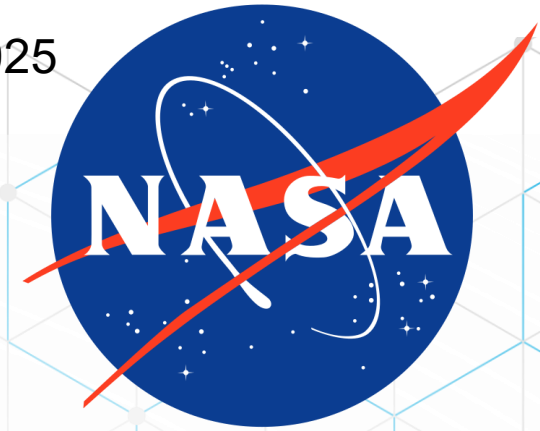
Nasa Formal Methods Symposium 2025
June 11th-13th 2025, Williamsburg, VA



Enforcing MAVLink Safety & Security Properties via Refined Multiparty Session Types

Arthur Amorim, Max Taylor, Trevor Kann, Gary T. Leavens, William L. Harrison, and Lance Joneckis.

Our approach uses runtime checks to enforce formal safety guarantees on unsafe system communications.

UAVs are increasingly being used in critical domains such as military operations.

JUN 3, 2025 6:00 AM ET

Ukraine's Drone Strikes Against Russia Could Become the Global Norm

The New York Times

Russia-Ukraine War › | The Latest | Photos | Ukraine's Drone Attack | Peace Talks | Troop Casualties

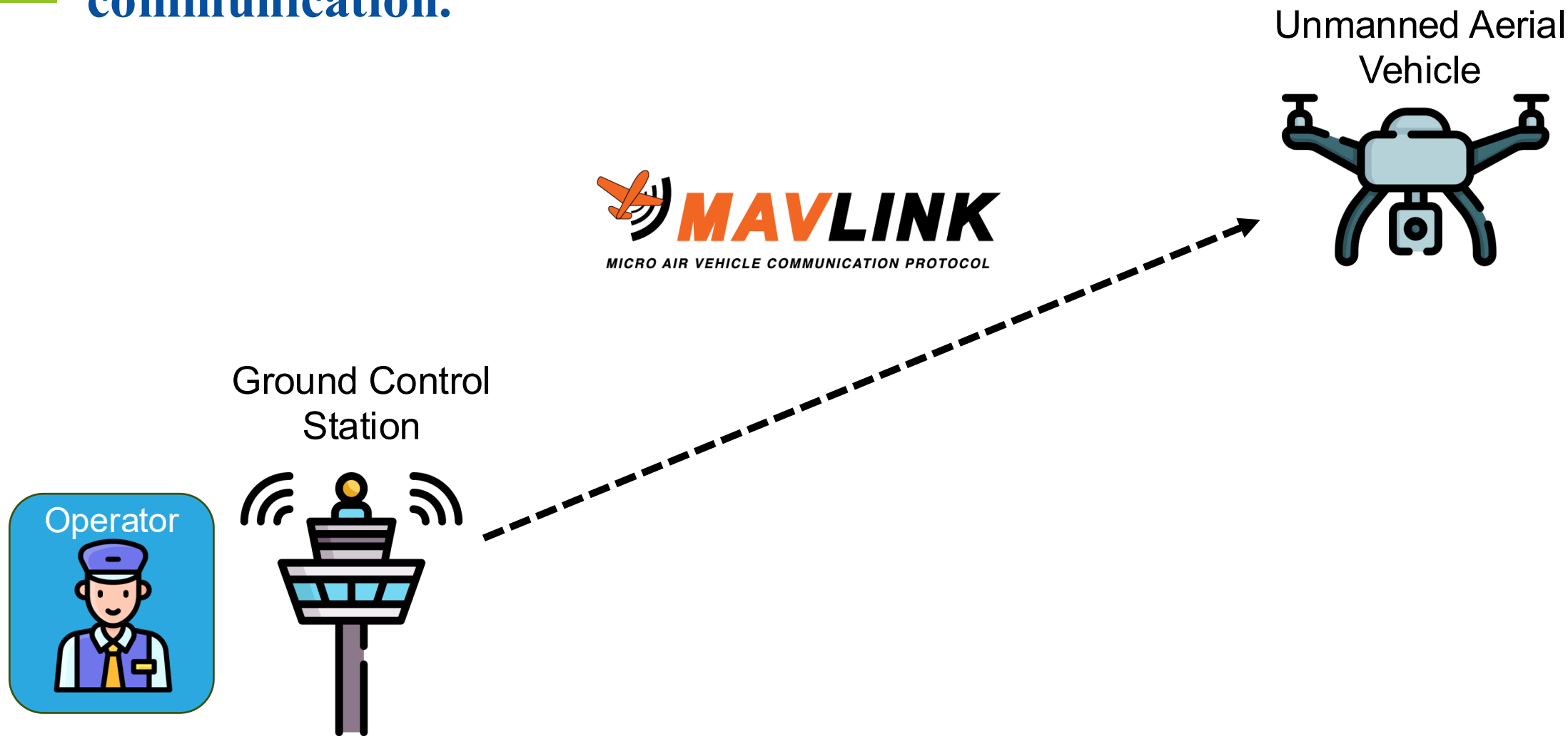
Ukraine Shows It Can Still Flip the Script on How Wars Are Waged

The attack demonstrated Ukraine's ability to use relatively cheap drones to take out expensive aircraft and to strike sites far from its borders.

Ukraine's Drone Swarms Are Destroying Russian Nuclear Bombers. What Happens Now?

Idaho National Laboratory

UAVs rely on protocols like MAVLink for mission-critical communication.

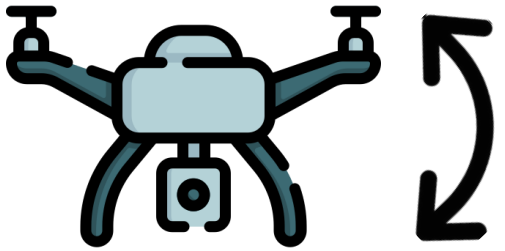


Stealthy attacks exploit commands that are allowed by the protocol but still induce undesired UAV behavior.

Known security shortcomings

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

Unmanned Aerial Vehicle



87 allowed but unsafe combinations of parameter values.

Ground Control Station

Attacker

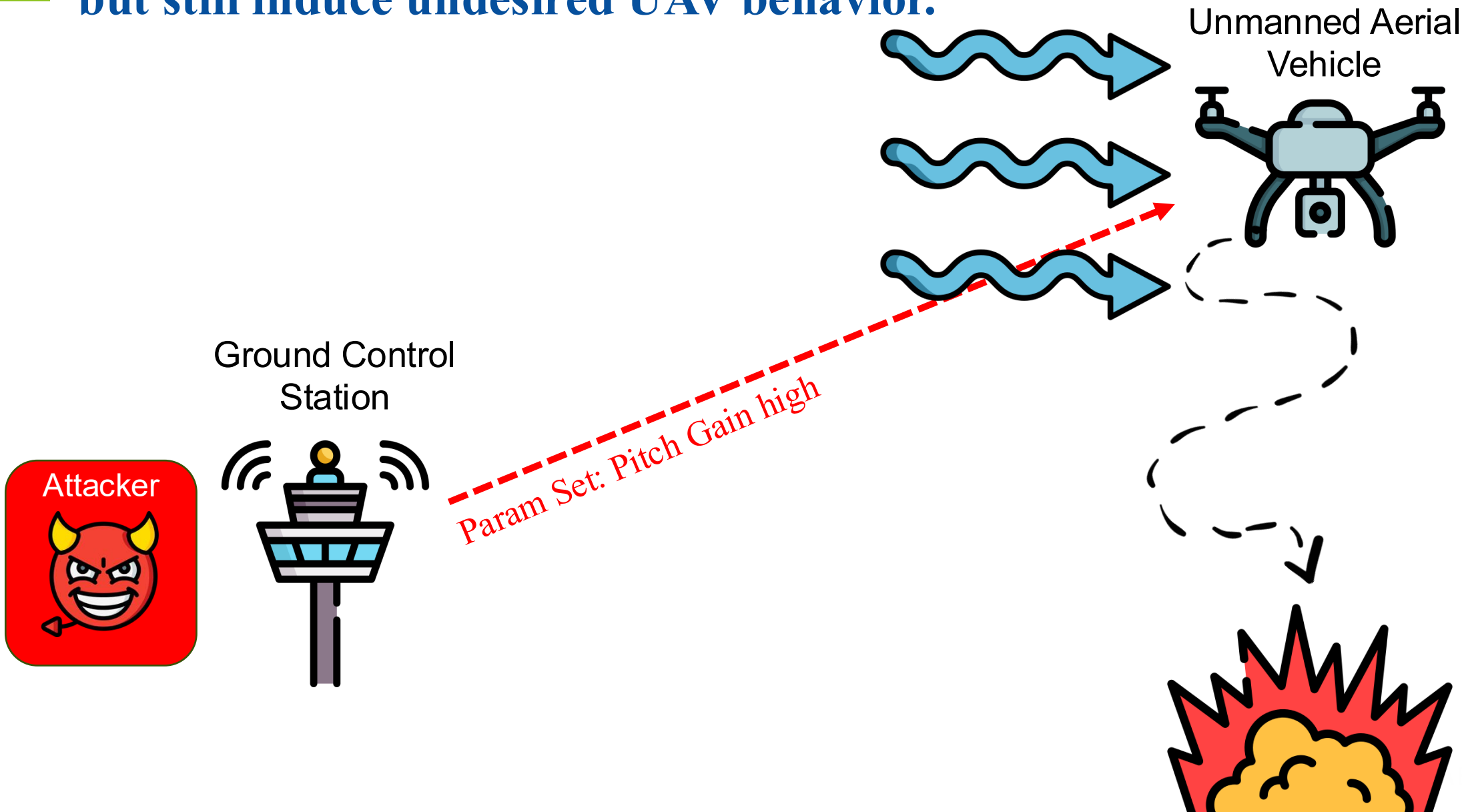
Param Set: Pitch Gain high

RVFUZZER: Finding Input Validation Bugs in Robotic Vehicles Through Control-Guided Testing

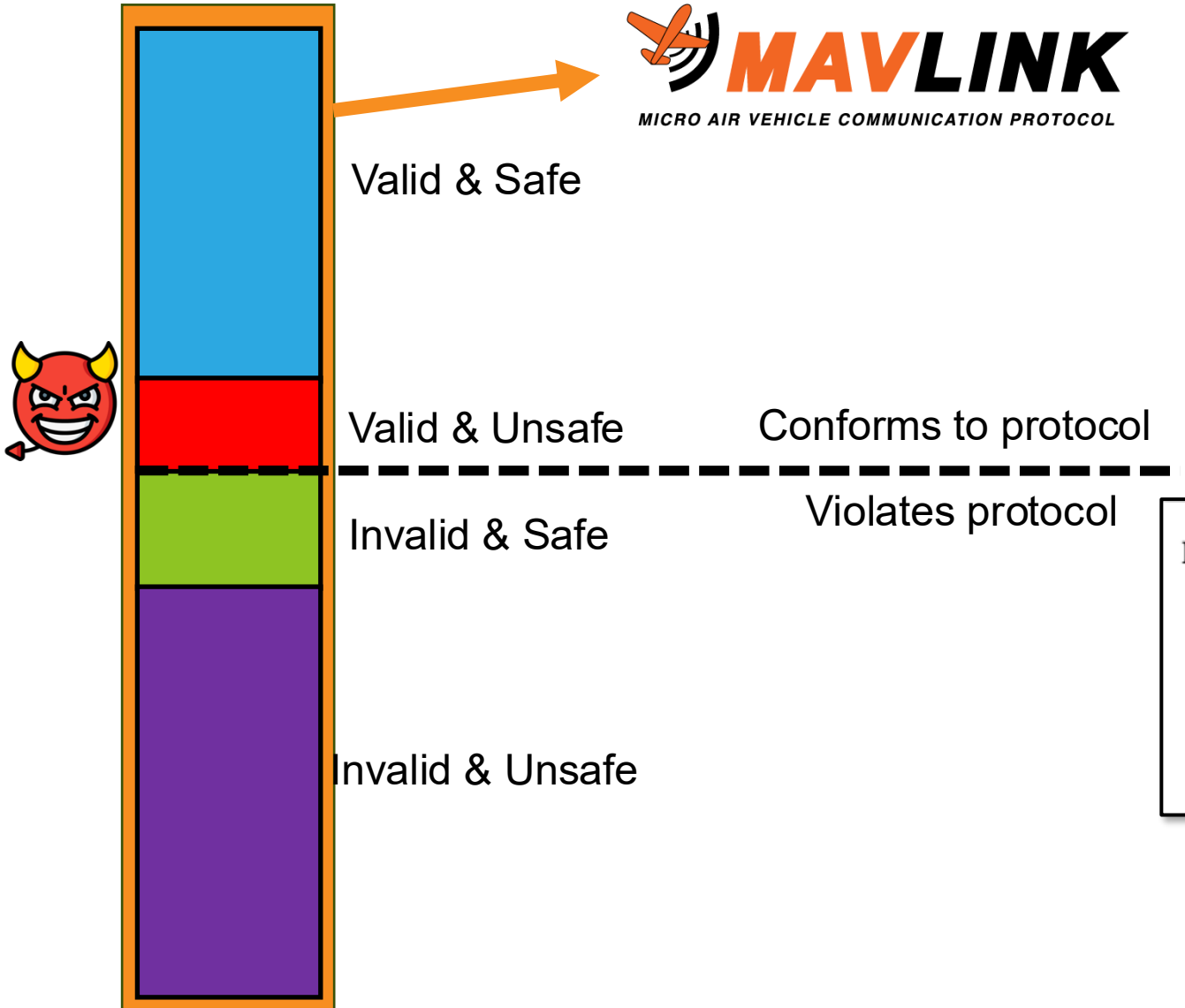
Taegyu Kim†, Chung Hwan Kim*, Junghwan Rhee*, Fan Fei†, Zhan Tu†, Gregory Walkup†
Xiangyu Zhang†, Xinyan Deng†, Dongyan Xu†
†*Purdue University, {tgkim, feif, tu17, gwalkup, xyzhang, xdeng, dxu}@purdue.edu*
**NEC Laboratories America, {chungkim, rhee}@nec-labs.com*

Source: https://www.usenix.org/system/files/sec19-kim.pdf

Stealthy attacks exploit commands that are allowed by the protocol but still induce undesired UAV behavior.



Stealthy attacks are very hard to detect by runtime monitoring because they exploit nominal behavior.



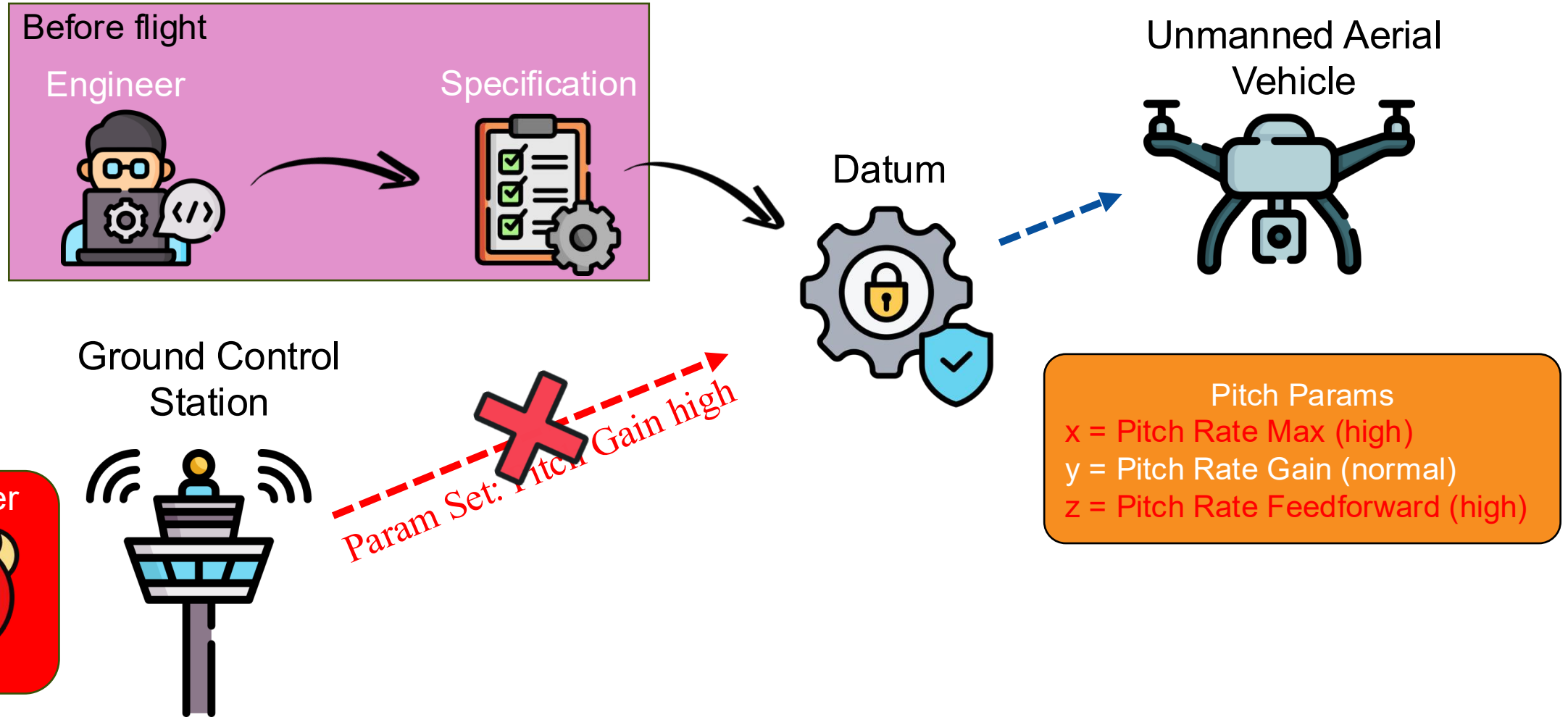
"Out of the remaining 87 bugs, the developers have so far independently confirmed 8 bugs and patched 7 of them."

RVFUZZER: Finding Input Validation Bugs in Robotic Vehicles Through Control-Guided Testing

Taegyu Kim†, Chung Hwan Kim*, Junghwan Rhee*, Fan Fei†, Zhan Tu†, Gregory Walkup†
Xiangyu Zhang†, Xinyan Deng†, Dongyan Xu†
†*Purdue University, {tgkim, feif, tu17, gwalkup, xyzhang, xdeng, dxu}@purdue.edu*
**NEC Laboratories America, {chungkim, rhee}@nec-labs.com*

Source: https://www.usenix.org/system/files/sec19-kim.pdf

Our runtime verification approach, Datum, can detect and prevent stealthy attacks at runtime by enforcing parameter-level constraints.



Datum is a framework that enhances the safety of protocols by writing protocol specifications with refinements.

1- Define protocol

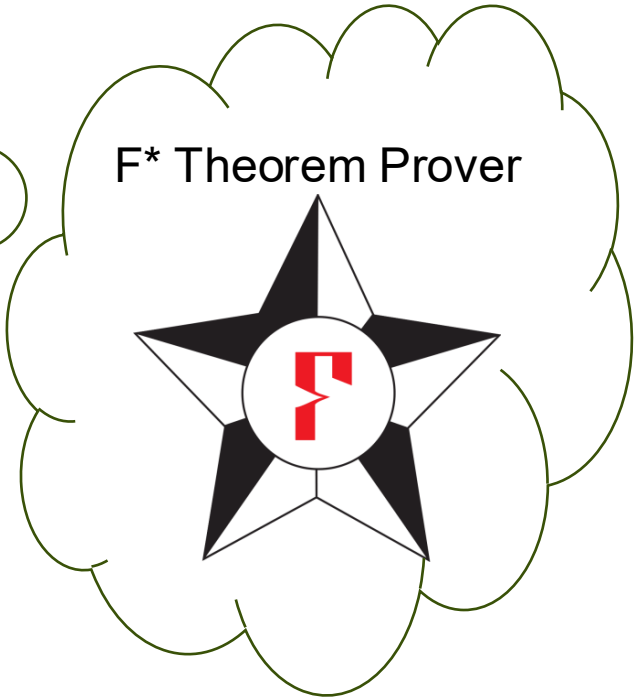
2- Add refinements

3- Check at runtime

Dynamically Assured Typed Universal Messaging

Datum

F* Theorem Prover



Idaho National Laboratory

F*'s strong dependent type system allows for the specification of safety constrains.

1- Define protocol

Common.xml

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

Generate Types

SCRIPT

F* theorem prover

```
<message name="MISSION_COUNT">
<field type="uint16_t"
name="count">
</field>
</message>"
```

```
type mission_count = {
count : uint16_t;
}
```

Idaho National Laboratory

Global refined multiparty session types (GRMPSTs) is the core logic underlying Datum.

2- Add refinements

Global Refined Multiparty Session Types

1- Refinements

$A \rightarrow B : MSG(x : \mathbb{N}\ \{x > 7\})$

2- Guarded Choice

$A \rightarrow B : (MSG1 : \mathbb{N}\ \{x = 10\}) \oplus (MSG2 : \mathbb{N}\ \{x \neq 10\})$

3 - Recursion

$\mu.T(n : \mathbb{N}\ \{0 \leq n \wedge n < 5\})\langle n = 0\rangle\ A \rightarrow B : MSG(y : Bool\ \{y = true\}).T\langle n = n + 1\rangle$

Idaho National Laboratory

Global refined multiparty session types (GRMPSTs) is the core logic underlying Datum.

2- Add refinements

Global Refined Multiparty Session Types

1- Refinements

$A \rightarrow B : MSG(x : \mathbb{N}\ \{x > 7\})$

2- Guarded Choice

$A \rightarrow B : (MSG1 : \mathbb{N}\ \{x = 10\}) \oplus (MSG2 : \mathbb{N}\ \{x \neq 10\})$

3 - Recursion

$\mu.T(n : \mathbb{N}\ \{0 \leq n \wedge n < 5\})\langle n = 0\rangle\ A \rightarrow B : MSG(y : Bool\ \{y = true\}).T\langle n = n + 1\rangle$

Idaho National Laboratory

Global refined multiparty session types (GRMPSTs) is the core logic underlying Datum.

2- Add refinements

Global Refined Multiparty Session Types

1- Refinements

$A \rightarrow B : MSG(x : \mathbb{N}\ \{x > 7\})$

2- Guarded Choice

$A \rightarrow B : (MSG1 : \mathbb{N}\ \{x = 10\}) \oplus (MSG2 : \mathbb{N}\ \{x \neq 10\})$

3 - Recursion

$\mu.T(n : \mathbb{N}\ \{0 \leq n \wedge n < 5\})\langle n = 0\rangle\ A \rightarrow B : MSG(y : Bool\ \{y = true\}).T\langle n = n + 1\rangle$

Conditional **Body** **Recur**

Idaho National Laboratory

Datum enforces inter-parameter constraints on MAVLink commands.

2- Add refinements

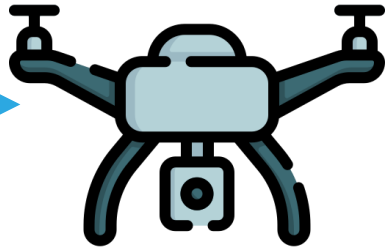
Check that the other pitch parameters have reasonable values

Datum

GCS

Param Set(Pitch Gain, n)

UAV



Idaho National Laboratory

Datum enforces inter-parameter constraints on MAVLink commands.

2- Add refinements

Existing specification

Added refinement

```
GCS → UAV: Param Set(Pitch Gain, n: ℕ {n < x*y*z*const})
```

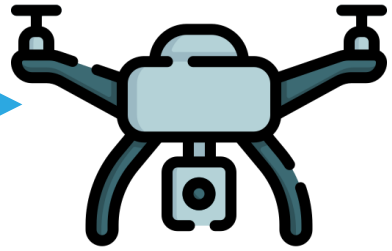
Pitch Parameter Values
x = Pitch Rate Max
y = Pitch Rate Gain
z = Pitch Rate Feedforward

Datum

GCS

Param Set(Pitch Gain, n)

UAV



Idaho National Laboratory

Datum enforces inter-parameter constraints on MAVLink commands.

3- Check at runtime

True

GCS → UAV: Param Set(Pitch Gain, n: ℕ {n < x*y*z*const})

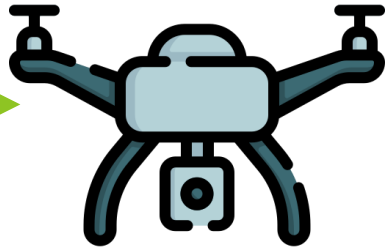
Pitch Parameter Values
x = Pitch Rate Max (normal)
y = Pitch Rate Gain (normal)
z = Pitch Rate Feedforward (low)

Datum

GCS

Param Set(Pitch Gain, n = high)

UAV



Idaho National Laboratory

Datum enforces inter-parameter constraints on MAVLink commands.

3- Check at runtime

False

GCS → UAV: Param Set(Pitch Gain, n: ℕ {n < x*y*z*const})

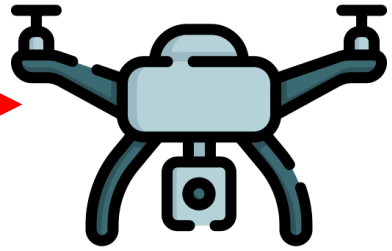
Pitch Parameter Values
x = Pitch Rate Max (high)
y = Pitch Rate Gain (normal)
z = Pitch Rate Feedforward (high)

Datum

GCS

Param Set(Pitch Gain, n = high)

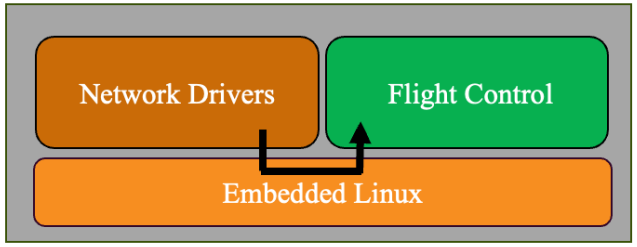
UAV



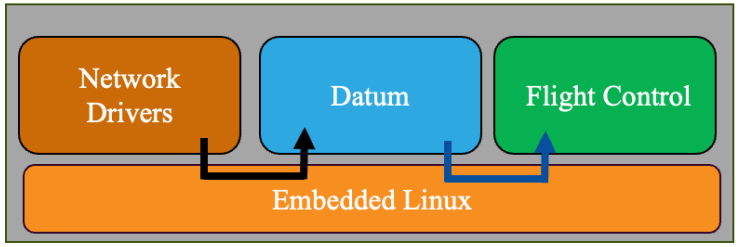
Idaho National Laboratory

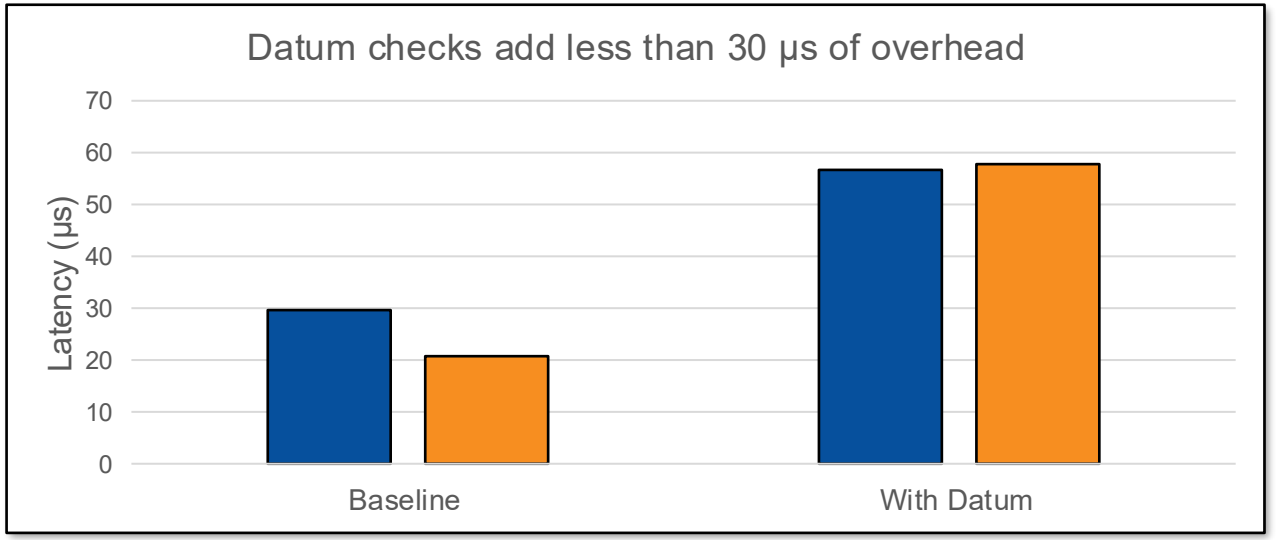
Off-the-shelf UAV software and SITL frameworks are fully operable with Datum

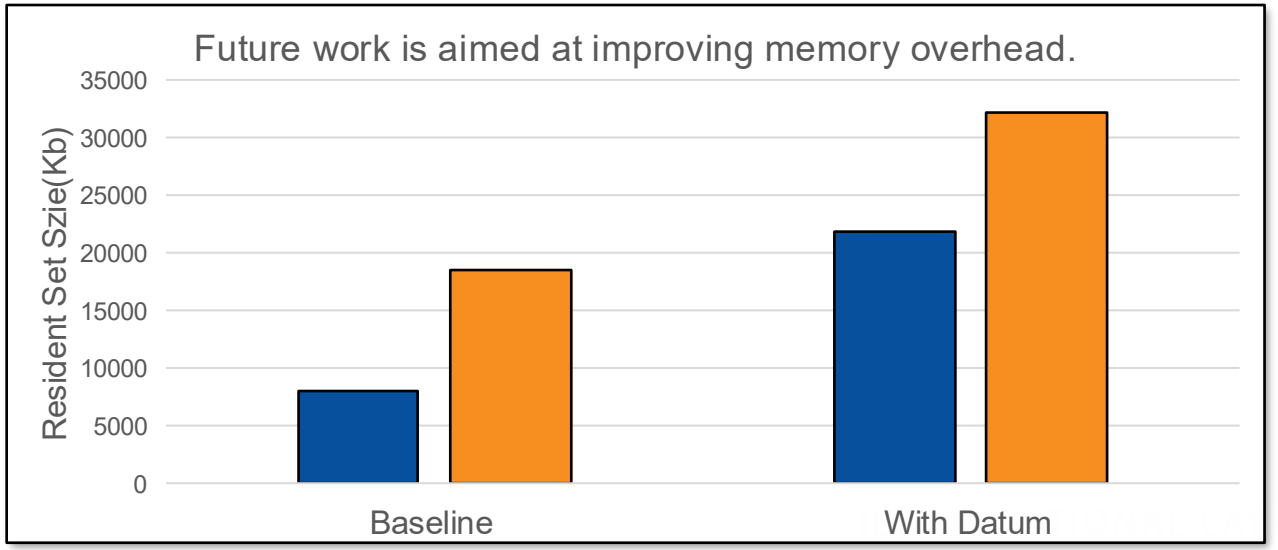
Baseline



With Datum







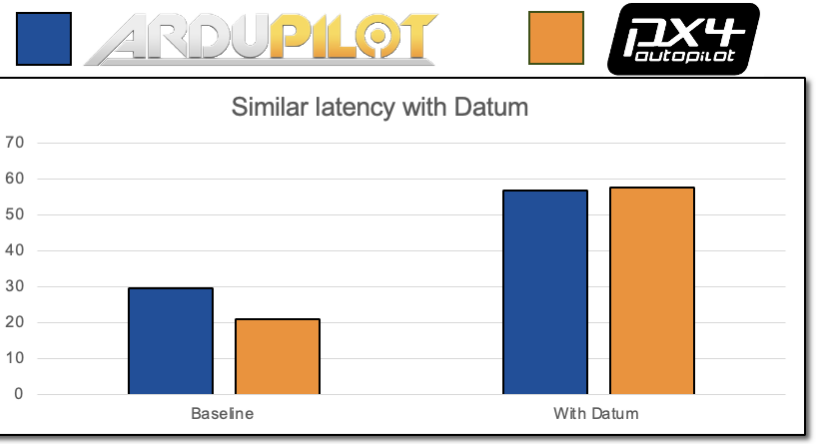
Idaho National Laboratory

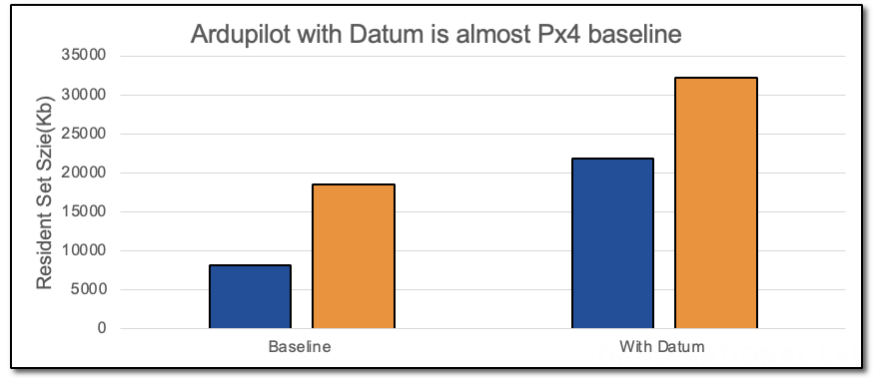
Our approach uses runtime checks to enforce formal safety guarantees on unsafe system communications.

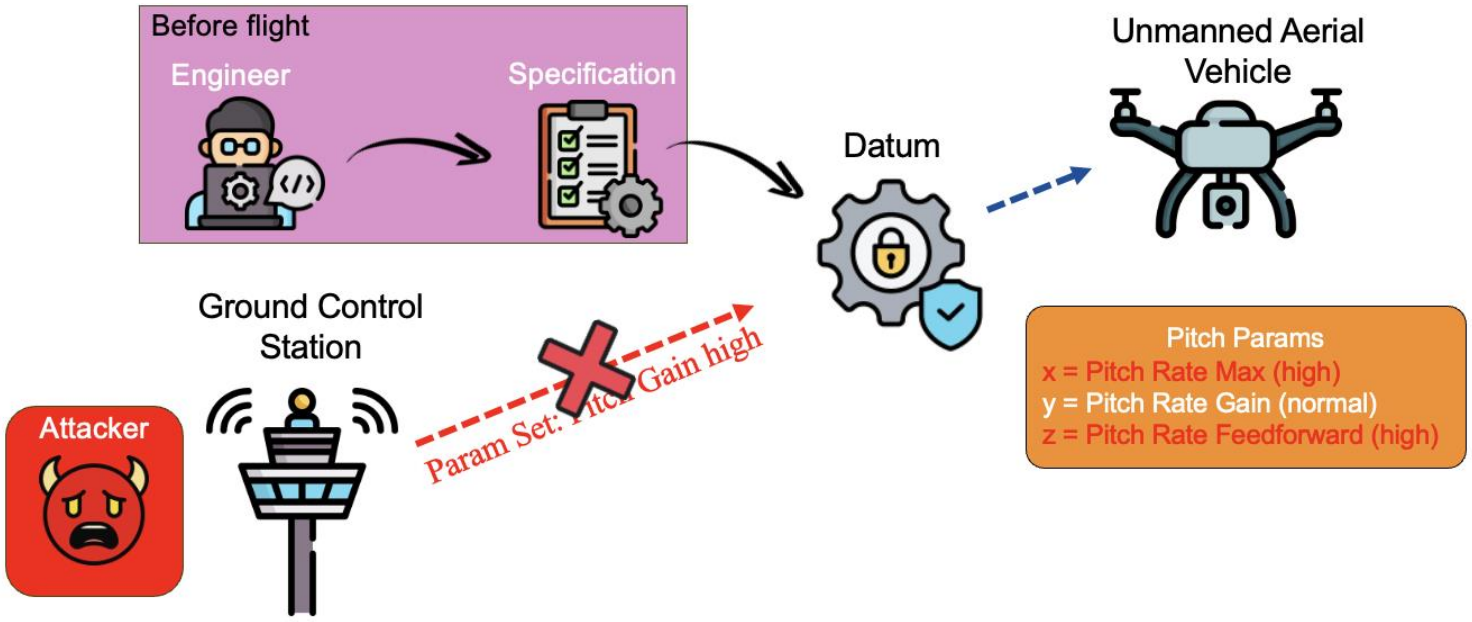
Dr. Gary T. Leavens
Arthur Amorim
Arthur.Amorim@ucf.edu
https://art-amorim.github.io/

Dr. Max Taylor
Dr. Lance Joneckis
Dr. Bill L. Harrison

Trevor Kann





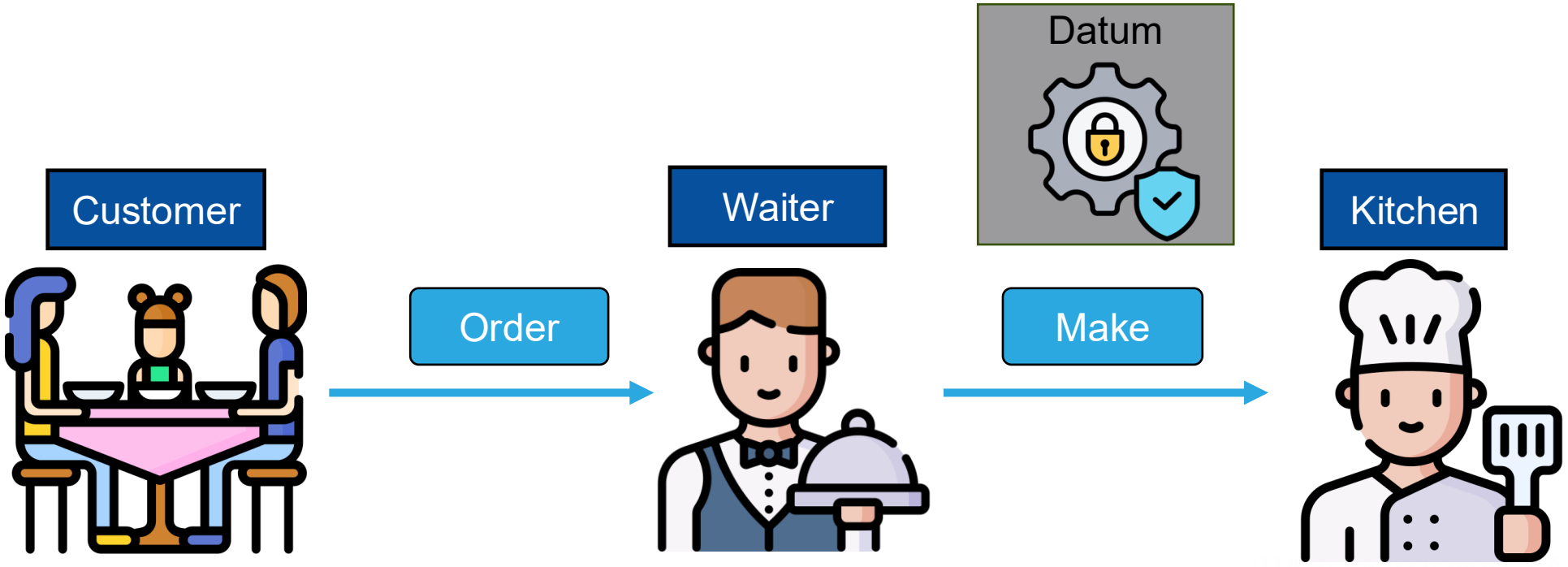


Idaho National Laboratory

GRMPSTs allow us to specify and monitor protocol-level safety properties using Datum.

2- Add refinements

Check: The Kitchen is making exactly the meal that was ordered



Idaho National Laboratory

GRMPSTs allow us to specify and monitor protocol-level safety properties using Datum.

2- Add refinements

Label

Refinement

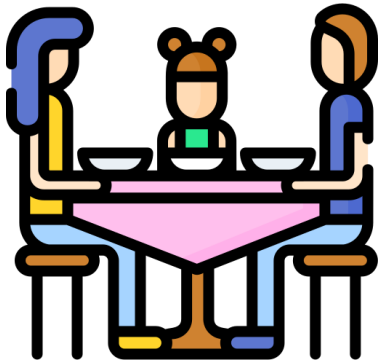
```
Customer → Waiter: Order(x: ℕ). Waiter → Kitchen: Make(y : ℕ{y = x})
```

Sender/receiver

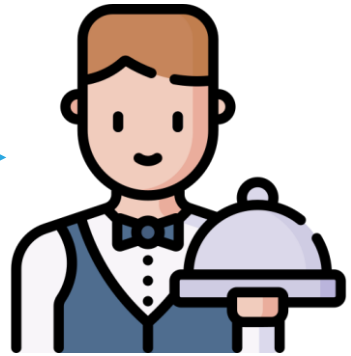
Payload

Datum

Customer



Waiter



Kitchen

Idaho National Laboratory

GRMPSTs allow us to specify and monitor protocol-level safety properties using Datum.

3- Runtime checks

Label

Refinement

```
Customer → Waiter: Order(x: ℕ). Waiter → Kitchen: Make(y : ℕ{y = x})
```

Sender/receiver

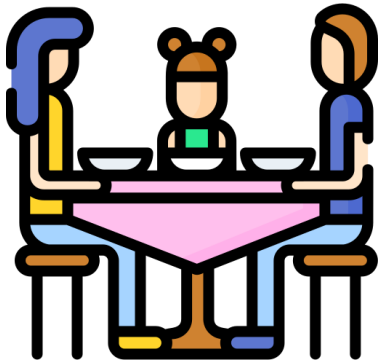
Payload

Datum

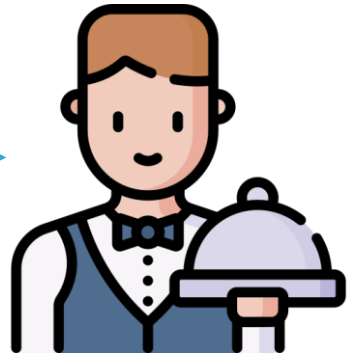
Customer

Waiter

Kitchen



Meal #7



Meal #7

Idaho National Laboratory

3- Runtime checks

GRMPSTs allow us to specify and monitor protocol-level safety properties using Datum.

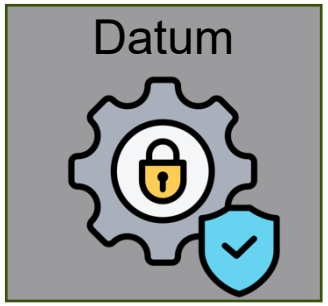
Label

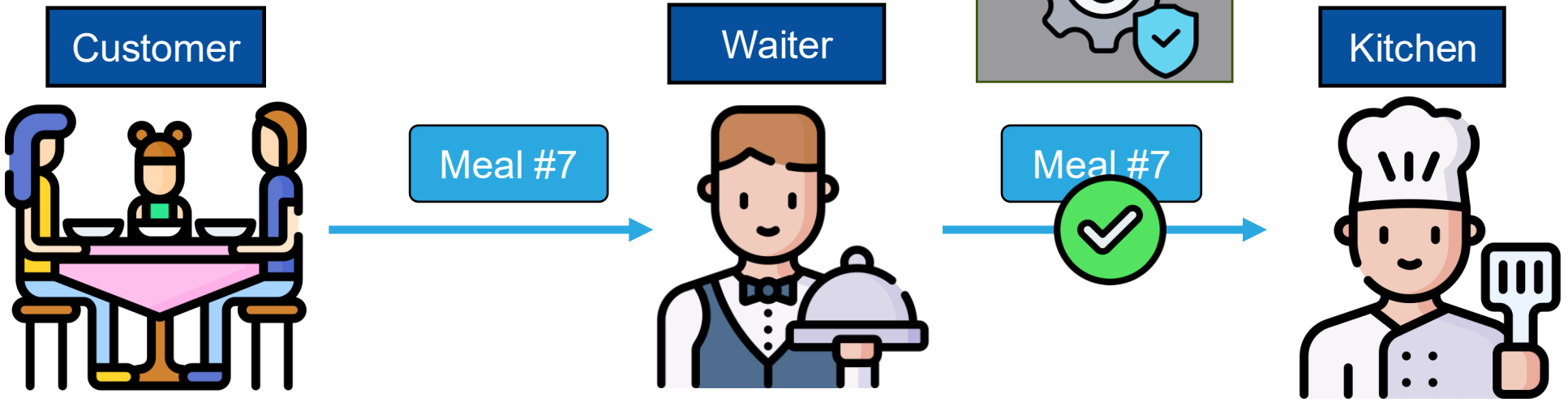
Refinement

```
Customer → Waiter: Order(x: ℕ). Waiter → Kitchen: Make(y : ℕ{y = x})
```

Sender/receiver

Payload





3- Runtime checks

GRMPSTs allow us to specify and monitor protocol-level safety properties using Datum.

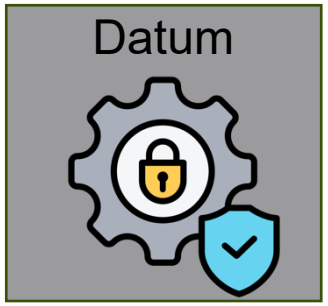
Label

Refinement

```
Customer → Waiter: Order(x: ℕ). Waiter → Kitchen: Make(y : ℕ{y = x})
```

Sender/receiver

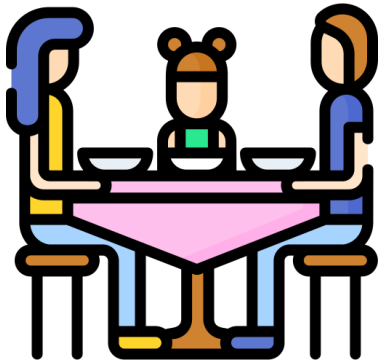
Payload



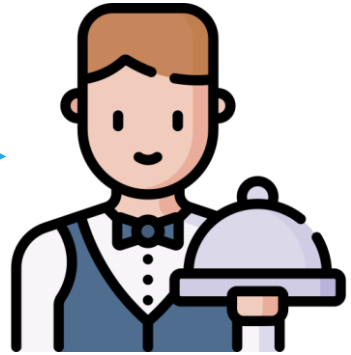
Customer

Waiter

Kitchen



Meal #7

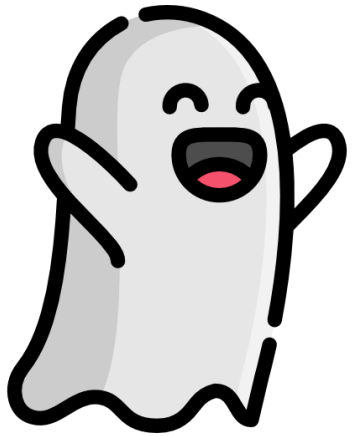


Meal #2

F*'s explicit handling of refinements

Native refinements

Hidden Monad



SMT integration