SCHEDULEAK

[RTAS 2019]

- Exfiltration of critical information
- Reconnaissance

"given knowledge of the scheduling algorithms used in the system, can we recreate its exact timing schedule?"

SYSTEM MODEL & REAL-TIME SCHEDULES

SYSTEM MODEL & REAL-TIME SCHEDULES

- **Consider three periodic real-time tasks**

	Period
1	5
2	6
3	15

SYSTEM MODEL & REAL-TIME SCHEDULES

- **Consider three periodic real-time tasks**
- **Their relative priorities are: 1 > 2 > 3**

	Period
1	5
2	6
3	15

SYSTEM MODEL & REAL-TIME SCHEDULES

- **Consider three periodic real-time tasks**
- **Their relative priorities are:** 1 > 2 > 3

	Period
1	5
2	6
3	15

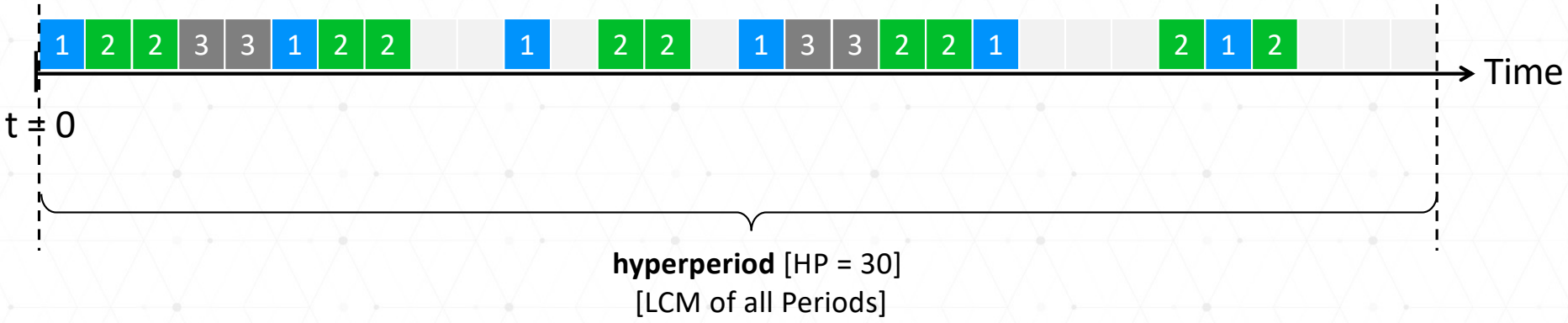
- **Their initial execution pattern would look like:**

SYSTEM MODEL & REAL-TIME SCHEDULES

- **Consider three periodic real-time tasks**
- **Their relative priorities are:** 1 > 2 > 3

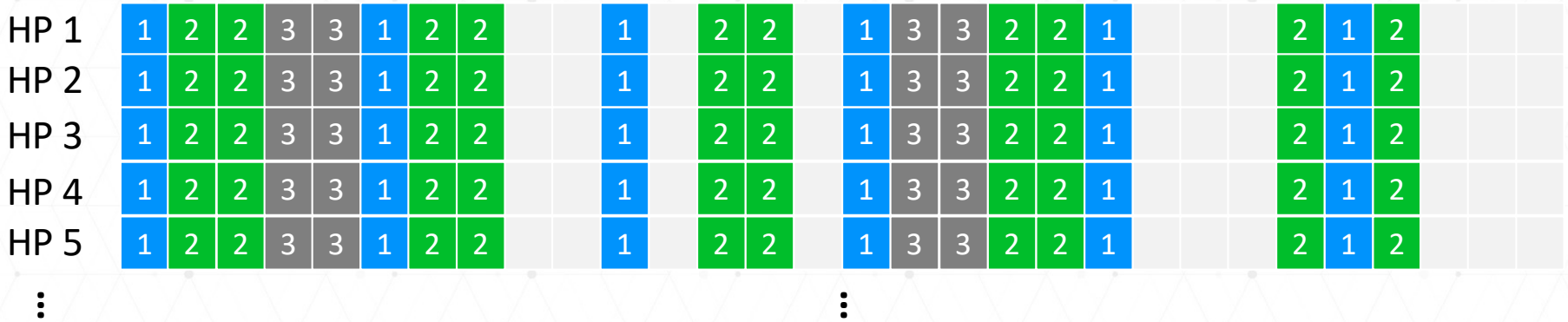
	Period
1	5
2	6
3	15

- **Their initial execution pattern would look like:**



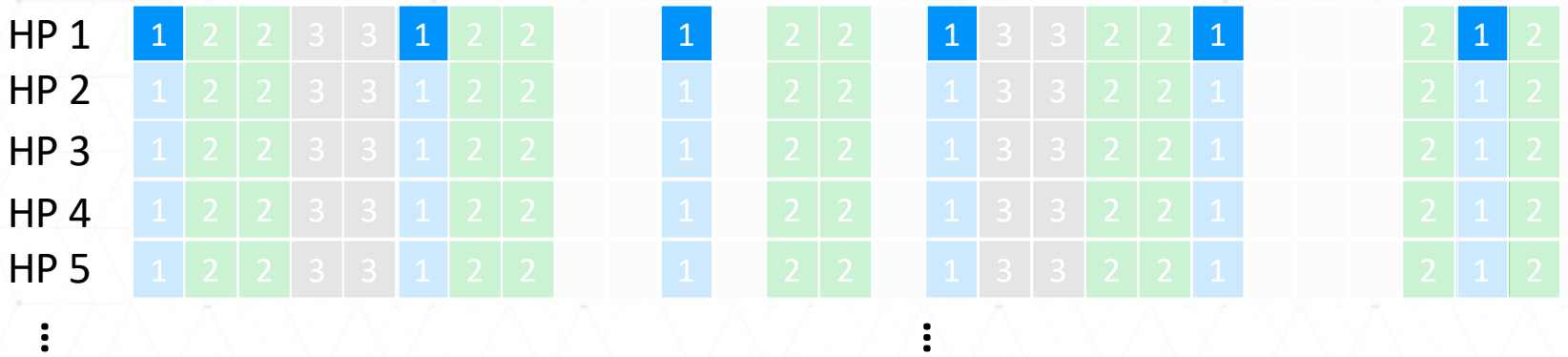
SYSTEM MODEL & REAL-TIME SCHEDULES

SYSTEM MODEL & REAL-TIME SCHEDULES



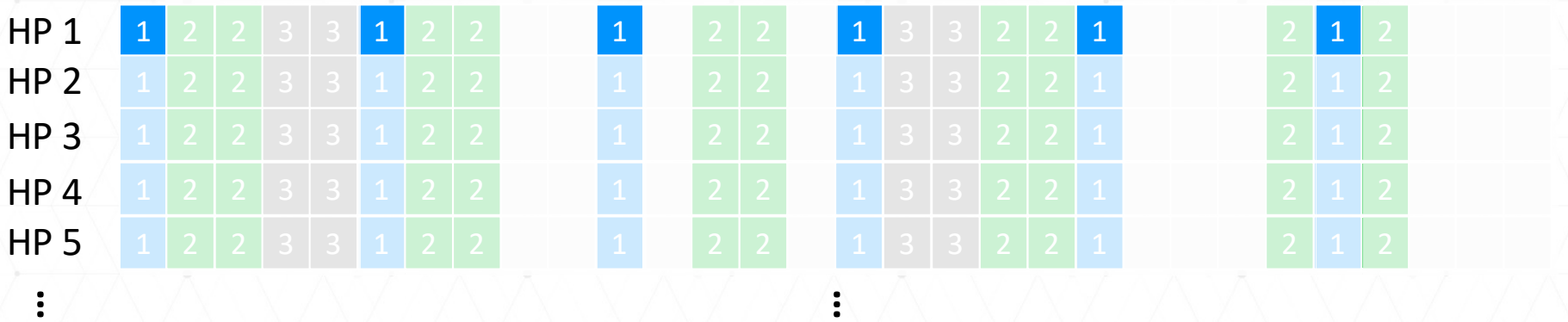
PROBLEM STATEMENT

PROBLEM STATEMENT



Can we predict **future** execution time points for critical task(s)? 1

PROBLEM STATEMENT



Can we predict **future** execution time points for critical task(s)? 1

PROBLEM STATEMENT

We use knowledge of the **regular execution patterns** in real-time systems

Can we predict **future** execution time points for critical task(s)? 1

PROBLEM STATEMENT

We use knowledge of the **regular execution patterns** in real-time systems

DEMONSTRATED A **CRITICAL SIDE CHANNEL** IN REAL-TIME SYSTEMS

Can we predict **future** execution time points for critical task(s)?

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Consider a UAV on a mission**

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Consider a UAV on a mission**
- **Takes [high-res] photos → points of interest [green]**
- **Camera → off or low-res mode otherwise**

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

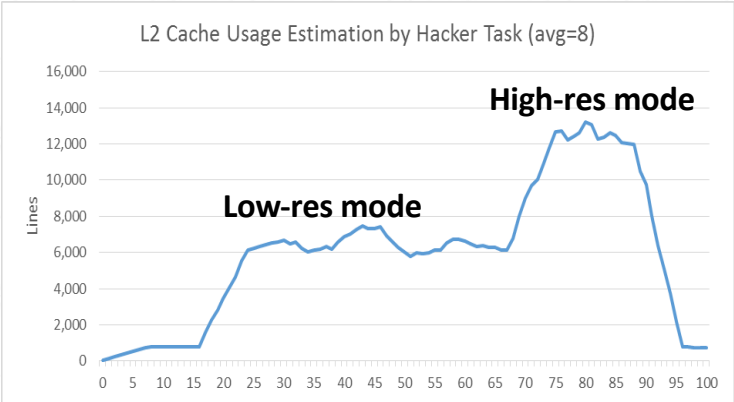
DEMONSTRATION 1

- **Attacker's goal**
  - **Recover location of interest points where memory usage [of victim] is high**

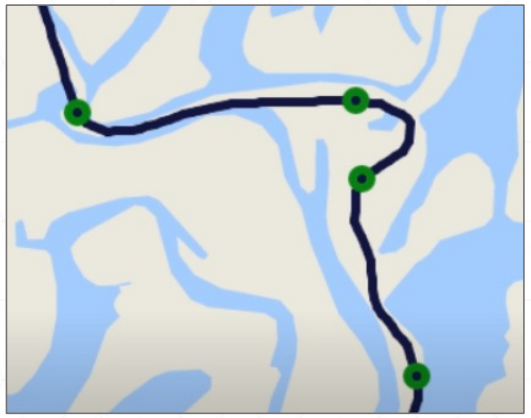
WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Attacker's goal**
  - **Recover location of interest points where memory usage [of victim] is high**



Measurements on Xilinx Zedboard Zynq-7000, FreeRTOS,
[CPU Freq: 666MHz, L2 Cache: 512KB, 32 byte line size]



true locations of interest

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Attacker's goal**
  - **Recover location of interest points where memory usage [of victim] is high**
- **Cache-timing side-channel attacks**

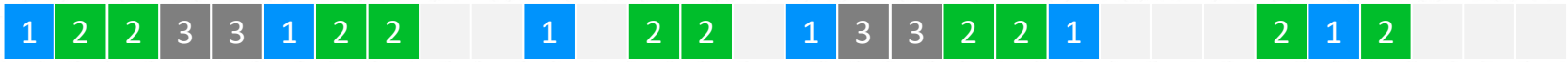
WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Attacker's goal**
  - **Recover location of interest points where memory usage [of victim] is high**
- **Cache-timing side-channel attacks**

Attacker: 2

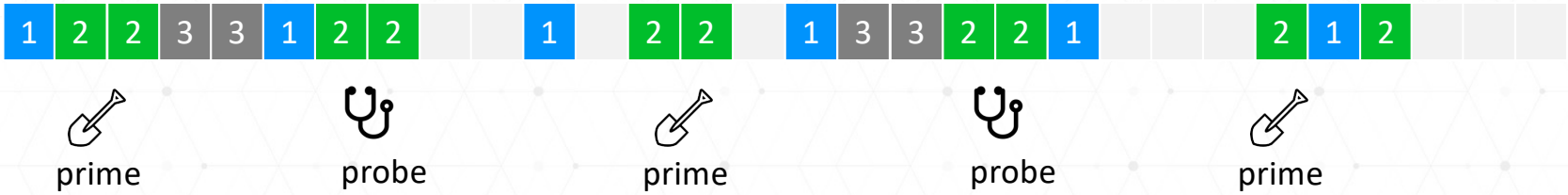
Target: 1 cache usage



WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- **Attacker's goal**
  - **Recover location of interest points where memory usage [of victim] is high**
- **Cache-timing side-channel attacks**



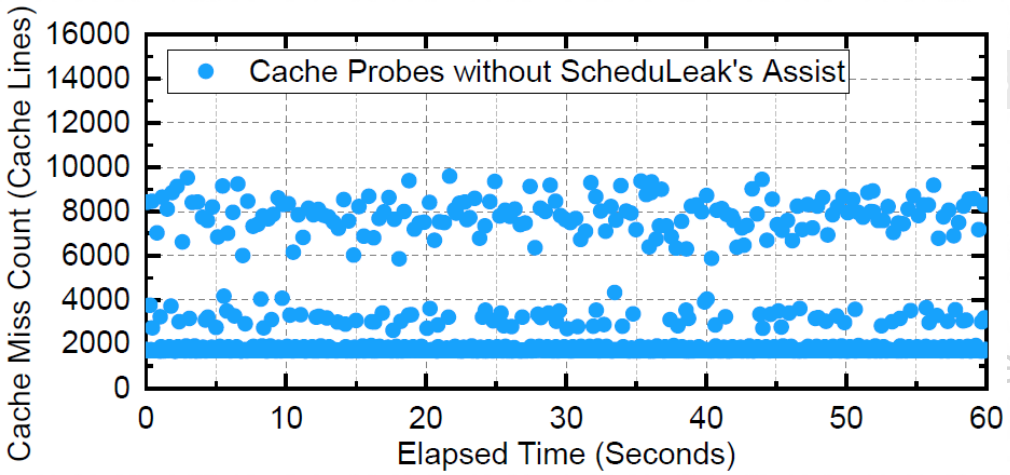
[Note: **no** information about future execution of victim task]

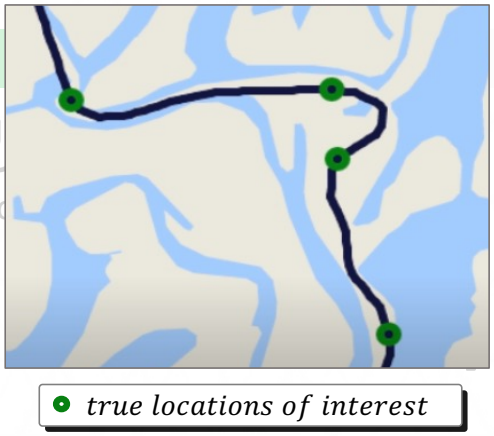
WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

DEMONSTRATION 1

- Attacker's goal
  - Recover location of interest points where memory usage [of victim] is high
- Cache-timing

Cache usage probes are indistinguishable



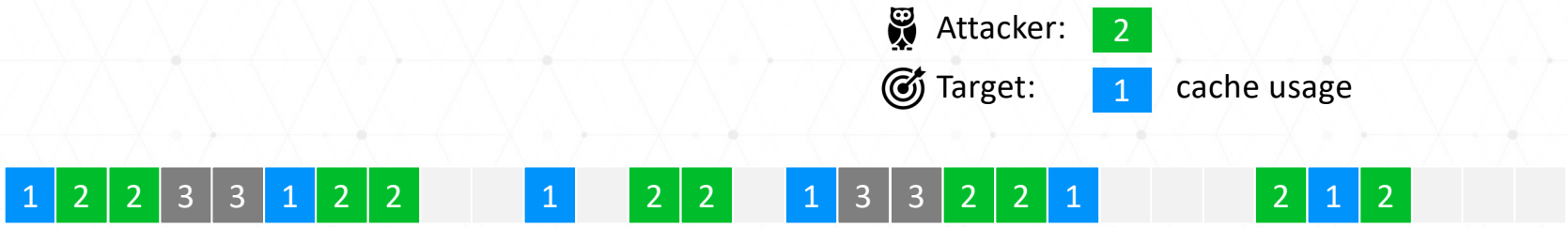


WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

- With precise timing information from the side-channel
  - Launch cache-timing attack at more precise points

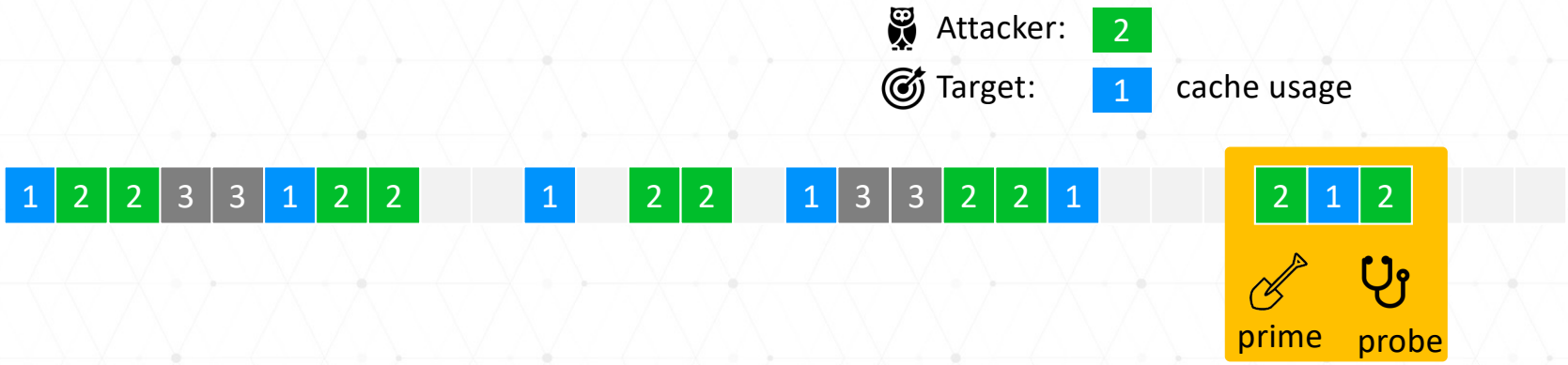
WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

- With precise timing information from the side-channel
  - Launch cache-timing attack at more precise points



WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

- **With precise timing information from the side-channel**
  - **Launch cache-timing attack at more precise points**

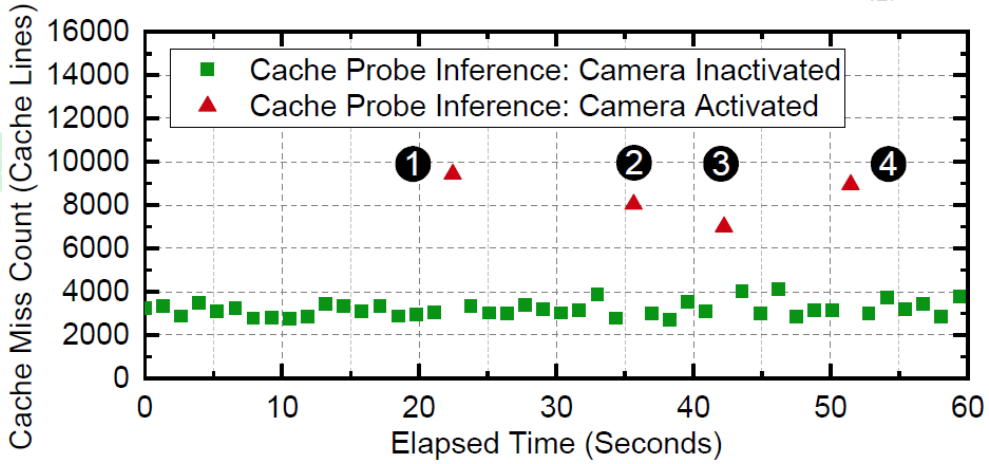


[Note: **very close** to the execution of victim task]

WHAT CAN WE DO WITH FUTURE EXECUTION INFORMATION?

- With precise timing information from the side-channel
  - Launch cache timing attack at more precise points

Four locations are **recovered** from cache usage probes



SYSTEM ASSUMPTIONS

Real-Time Tasks

- **Periodic**
  - Jobs released periodically
  - Relative deadlines
- **Sporadic**
  - Release/arrival times specified
  - Inter-arrival times
  - Absolute deadlines

worst-case execution times

SYSTEM ASSUMPTIONS

- **Assumption: Fixed-Priority Real-Time Systems [E.g. RM]**
  - **Attacker's task (observer task)** *periodic* or *sporadic*
  - **Victim task** *periodic*
  - **Other tasks** *periodic* or *sporadic*

Real-Time Tasks

- **Periodic**
  - Jobs released periodically
  - Relative deadlines
- **Sporadic**
  - Release/arrival times specified
  - Inter-arrival times
  - Absolute deadlines

worst-case execution times

SYSTEM ASSUMPTIONS

- **Assumption: Fixed-Priority Real-Time Systems [E.g. RM]**
  - **Attacker's task (observer task)** *periodic* or *sporadic*
  - **Victim task** *periodic*
  - **Other tasks** *periodic* or *sporadic*

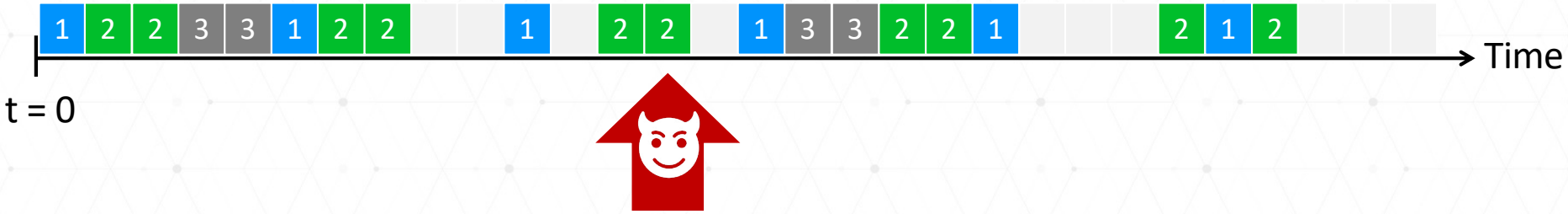
- **Requirements**
  - **The attacker knows the victim task's period**
  - **The observer task has lower priority than the victim task**

Real-Time Tasks

- **Periodic**
  - Jobs released periodically
  - Relative deadlines
- **Sporadic**
  - Release/arrival times specified
  - Inter-arrival times
  - Absolute deadlines

worst-case execution times

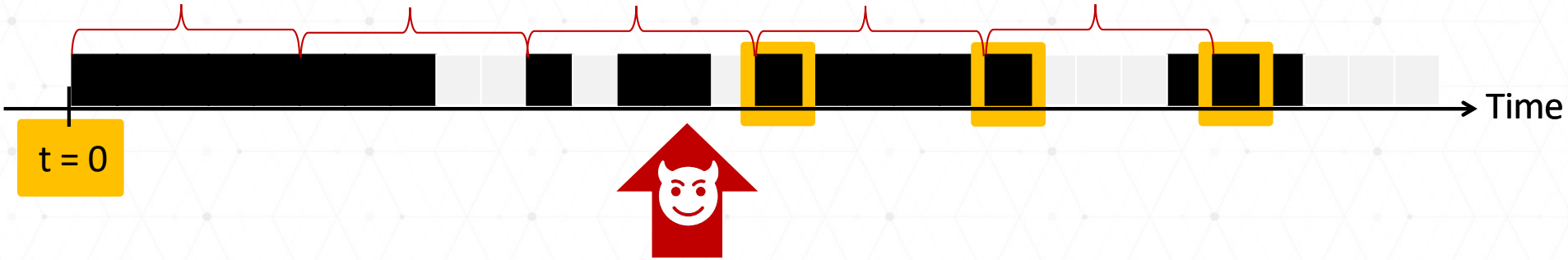
WHY IS THIS HARD?



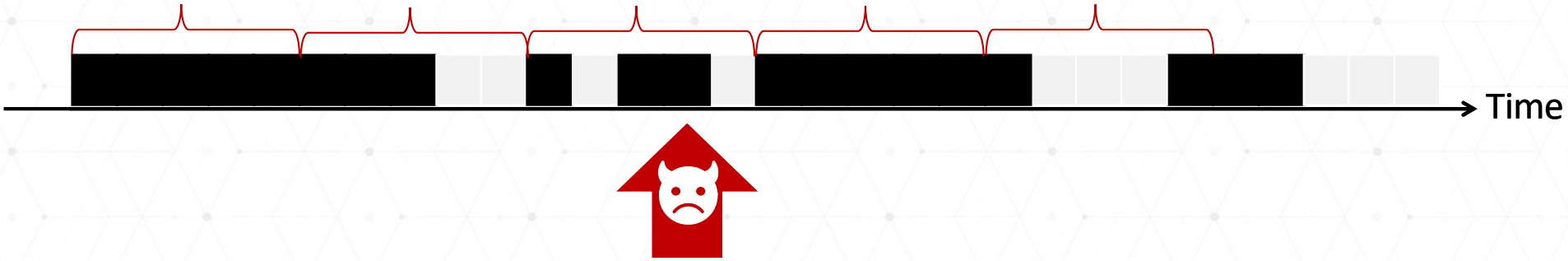
WHY IS THIS HARD?

WHY IS THIS HARD?

WHY IS THIS HARD?

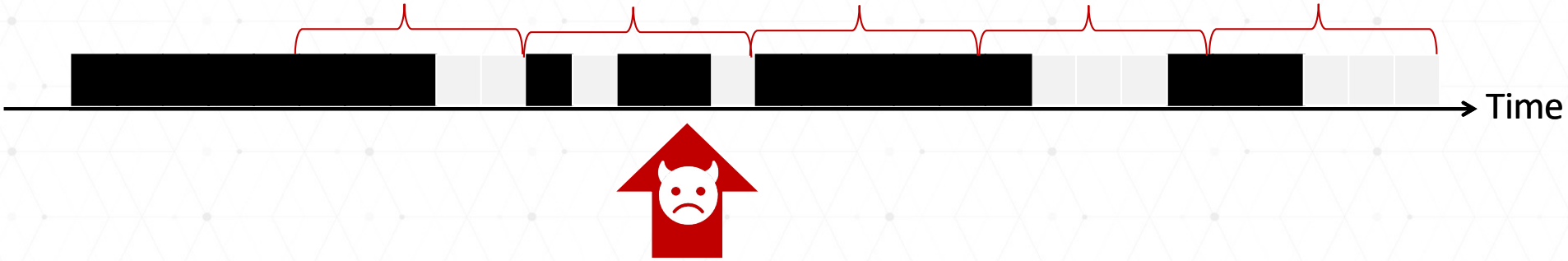


WHY IS THIS HARD?



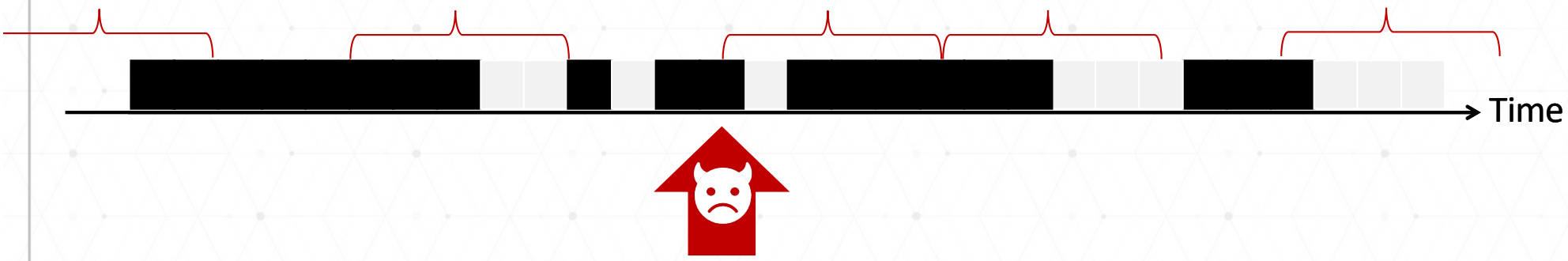
- Attacker enters system at random point in time

WHY IS THIS HARD?



- Attacker enters system at random point in time

WHY IS THIS HARD?



- Attacker enters system at random point in time

WHY IS THIS HARD?

- Attacker enters system at random point in time
- Maybe look at the scheduler?

WHY IS THIS HARD?

- Attacker enters system at random point in time
- Maybe look at the scheduler? → need to break user/kernel boundary

WHY IS THIS HARD?

- Attacker enters system at random point in time
- Maybe look at the scheduler? → need to break user/kernel boundary
- Attacker wants to stay undetected

ATTACK SCENARIO OVERVIEW

There is some schedule (on the victim system)

ATTACK SCENARIO OVERVIEW

There is some schedule (on the victim system)

The adversary **observes** and **analyzes** the schedule and **reconstructs** precise timing information

Inferring arrivals of a “victim” task

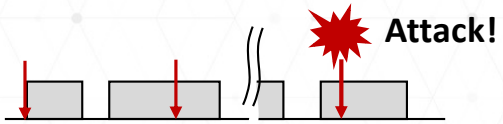
ATTACK SCENARIO OVERVIEW

There is some schedule (on the victim system)

The adversary **observes** and **analyzes** the schedule and **reconstructs** precise timing information

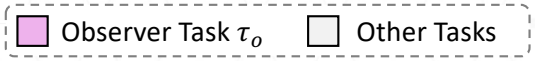
Inferring arrivals of a "victim" task

The attacker can then launch a major attack at a future instant that can **cause the most amount of damage**



SCHEDULEAK ALGORITHMS

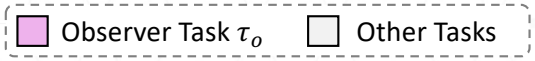
Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



SCHEDULEAK ALGORITHMS

Observer task has **lower priority** than victim task

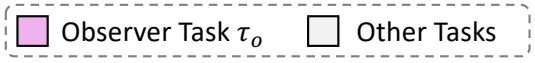
Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



SCHEDULEAK ALGORITHMS

1. Reconstruct execution intervals of τ_v

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1

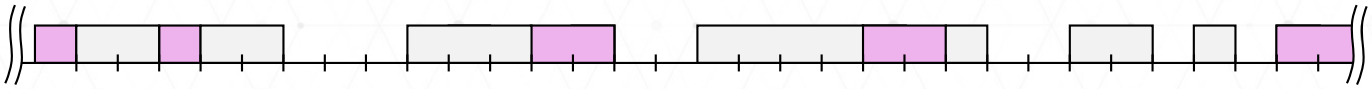


SCHEDULEAK ALGORITHMS

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1

1. Reconstruct execution intervals of τ_v

System Schedule Ground Truth:



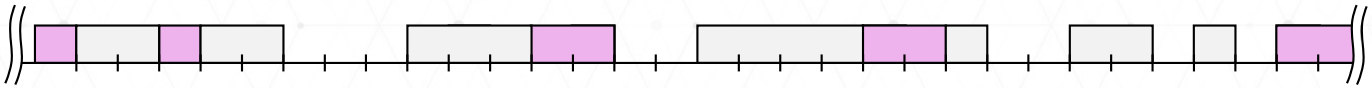
Observer Task τ_o Other Tasks

SCHEDULEAK ALGORITHMS

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1

1 Reconstruct execution intervals of τ_v

System Schedule Ground Truth:



What the attacker can observe

Execution Intervals Reconstructed by the Observer Task:

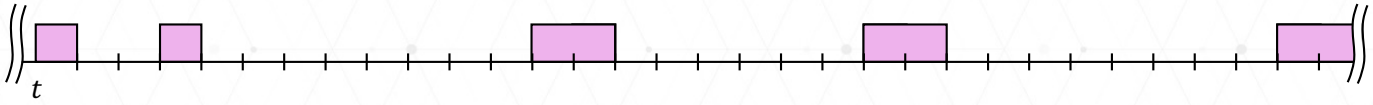
Some tasks preempted the observer task

Observer Task τ_o Other Tasks

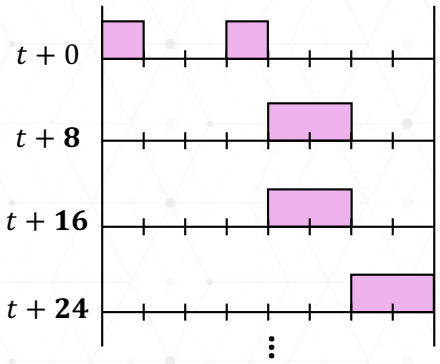
SCHEDULEAK ALGORITHMS

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1

2 Organize the execution intervals in a "**schedule ladder diagram**"



Place the intervals in a **ladder diagram** (width equals the victim task's period)



Still dealing with the Observer task executions

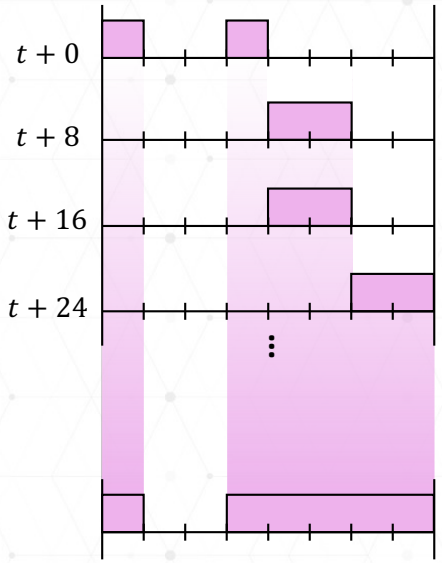
SCHEDULEAK ALGORITHMS

2. Organize the execution intervals in a "**schedule ladder diagram**"

   ↓

   Take union of the execution intervals

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



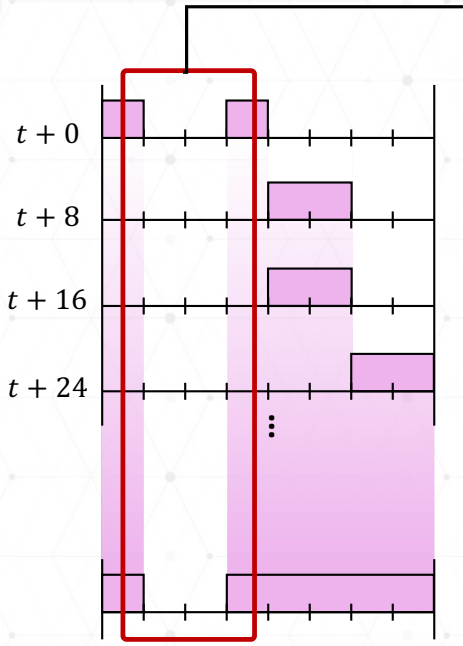
SCHEDULEAK ALGORITHMS

2 Organize the execution intervals in a "**schedule ladder diagram**"

↓

Take union of the execution intervals

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



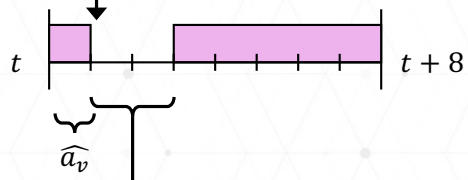
TASKS WITH LOWER PRIORITIES (E.G. OBSERVER TASK) **CANNOT** APPEAR IN THIS COLUMN!

SCHEDULEAK ALGORITHMS

3 Infer the victim task's initial offset

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1

We take the **starting point** of the empty column as the inference of the victim task's initial offset



TASKS WITH LOWER PRIORITIES (E.G. OBSERVER TASK) **CANNOT** APPEAR IN THIS COLUMN!

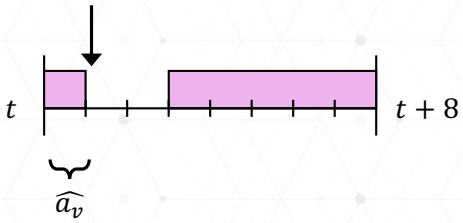
SCHEDULEAK ALGORITHMS

3 Infer the victim task's initial offset

↓

Predict the victim task's future arrivals

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



The **victim task's future arrival times** can be computed by

$$t + \widehat{a_v} + p_v * T$$

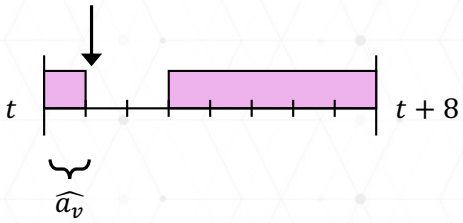
SCHEDULEAK ALGORITHMS

3 Infer the victim task's initial offset

↓

Predict the victim task's future arrivals

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



The **victim task's future arrival times** can be computed by

$$t + \widehat{a_v} + p_v * T$$

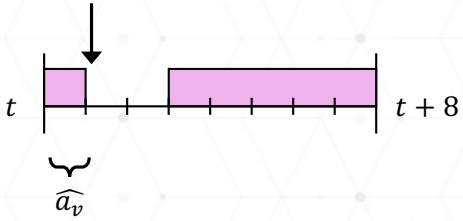
- t: ladder diagram starting point
- $\widehat{a_v}$: Inferred victim task's Initial offset
- p_v: victim task's period
- T: desired arrival number

SCHEDULEAK ALGORITHMS

3 Infer the victim task's initial offset

Predict the victim task's future arrivals

Task ID	Period	Exec Time
Observer Task	15	1
Task 2	10	2
Victim Task (τ_v)	8	2
Task 4	6	1



The **victim task's future arrival times** can be computed by

$$t + \widehat{a_v} + p_v * T$$

- t: ladder diagram starting point
- $\widehat{a_v}$: Inferred victim task's Initial offset
- p_v: victim task's period
- T: desired arrival number

CAN PREDICT, WITH HIGH PRECISION, **FUTURE** ARRIVAL TIMES OF VICTIM!

PERFORMANCE EVALUATION

- **Synthetic Task Sets**

6000 Task Sets:

Task Set Utilization

[0.01,0.1) ... [0.91, 1.0)

10 groups

×

The Number of Tasks

5, 7, 9, 11, 13, 15

6 groups

× **100**

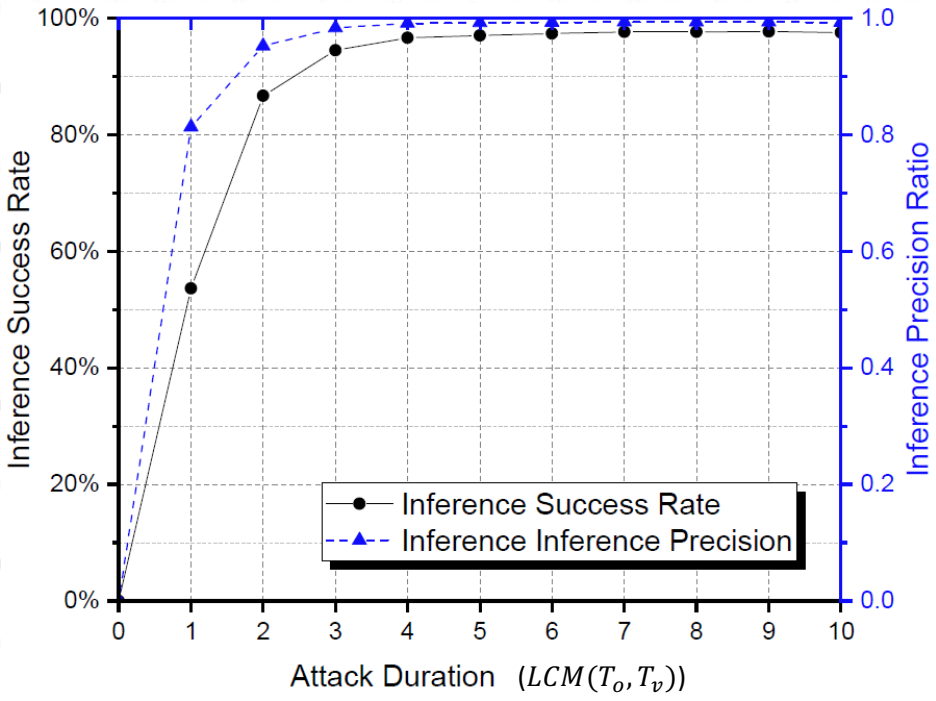
PERFORMANCE EVALUATION: METRICS

- **Inference Precision Ratio**
  the ratio of how close the inference to the true task starting point

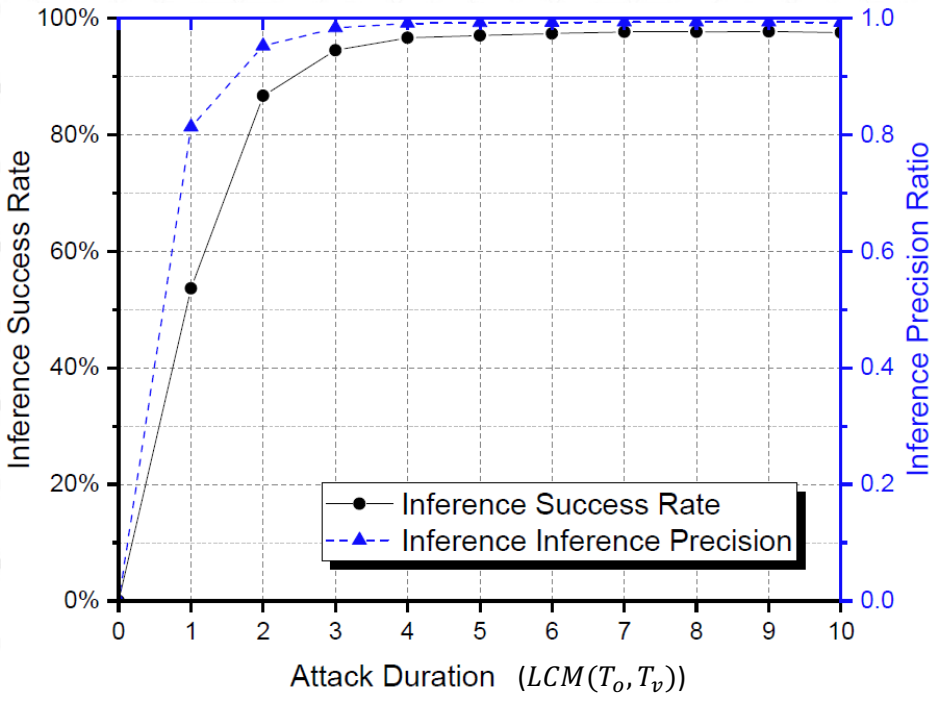
PERFORMANCE EVALUATION: METRICS

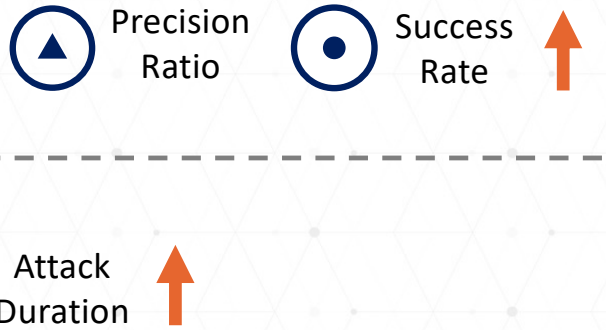
- **Inference Precision Ratio**
  the ratio of how close the inference to the true task starting point
- **Inference Success Rate**
  an inference is successful if attacker can exactly infer the starting point of the victim task

PERFORMANCE EVALUATION: RESULTS

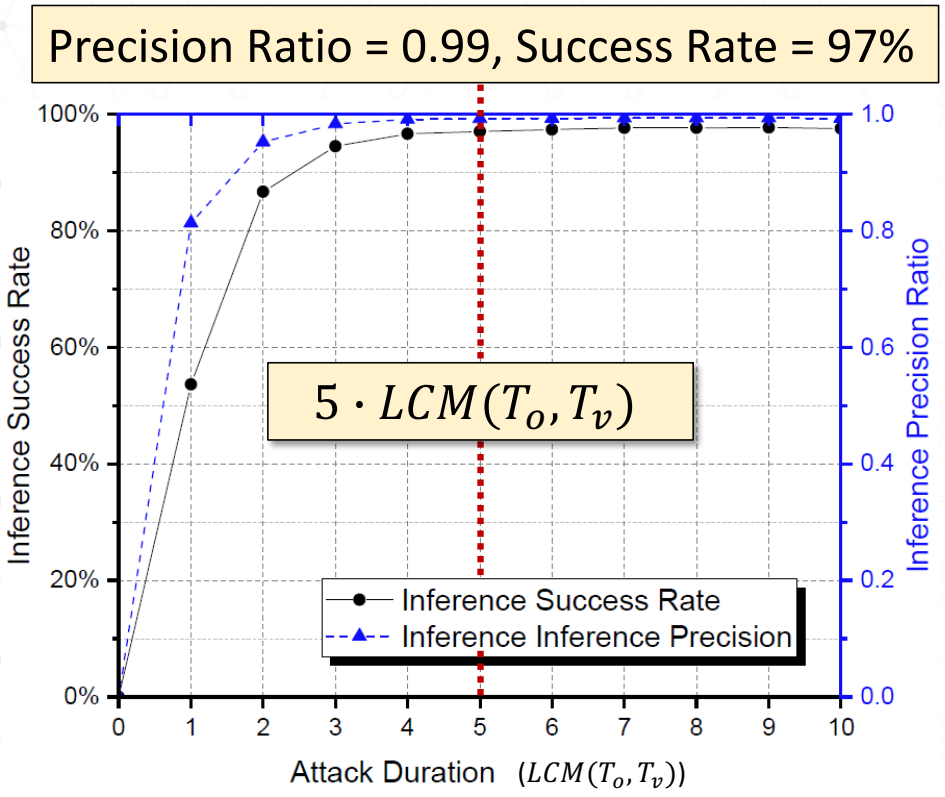


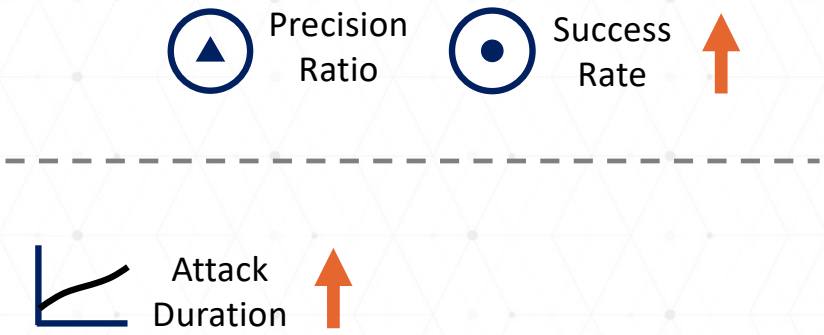
PERFORMANCE EVALUATION: RESULTS



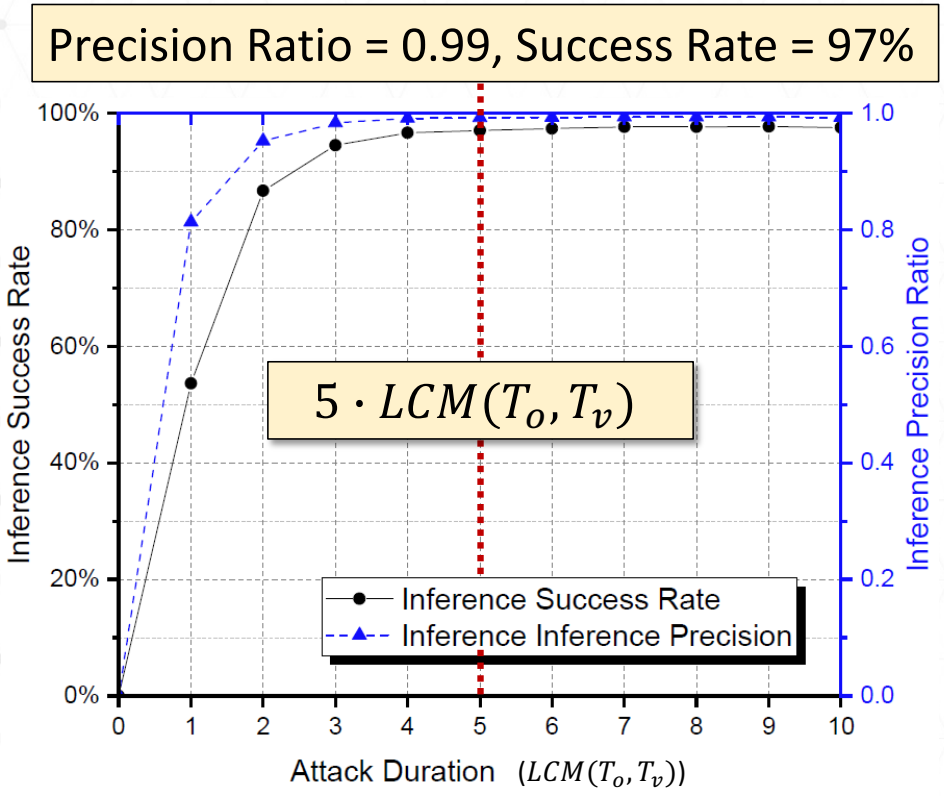


PERFORMANCE EVALUATION: RESULTS





PERFORMANCE EVALUATION: RESULTS



Precision Ratio

Success Rate

Attack Duration

The Number of Tasks

Coverage Ratio

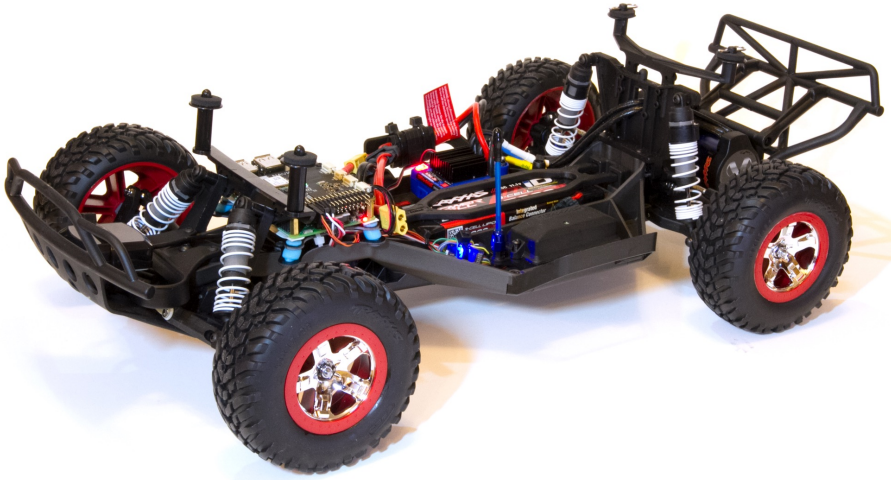
Sporadic Task Ratio

Task Set Utilization

Observer Task Priority

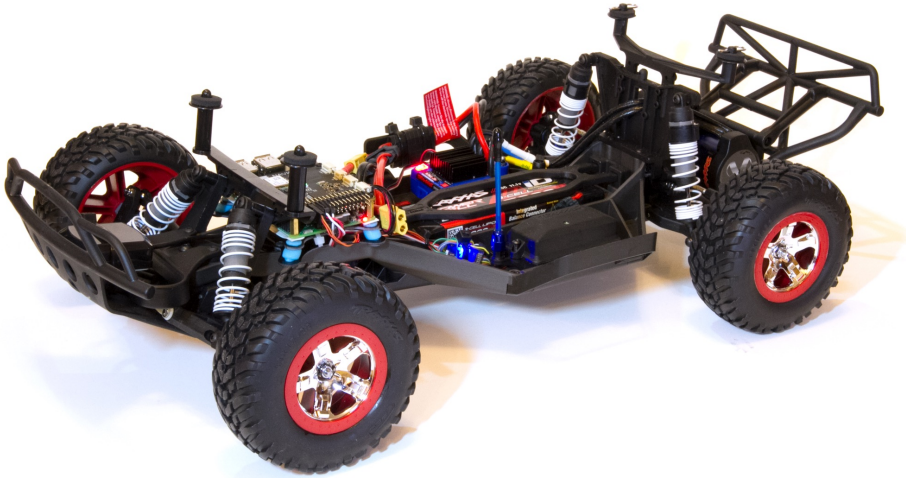
WHAT CAN WE DO WITH INFORMATION GLEANED USING SCHEDULEAK?

DEMONSTRATION 2



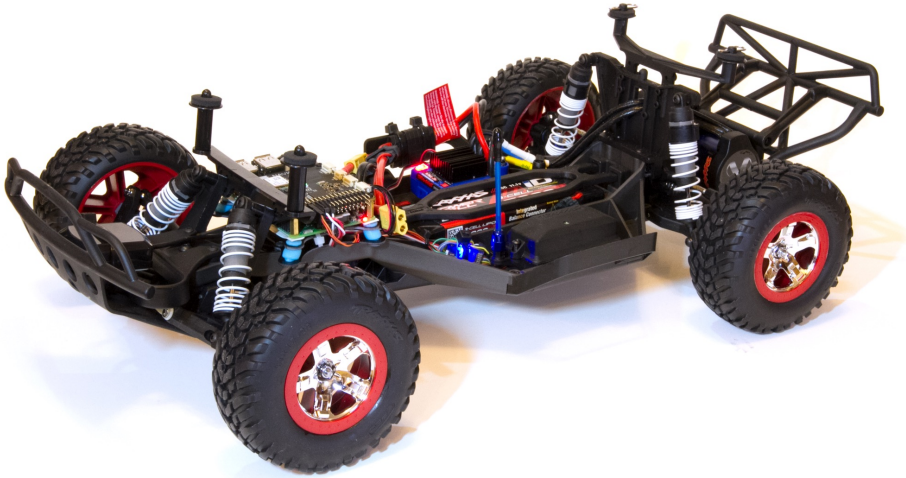
DEMONSTRATION 2

- **Imagine attacker controls a real-time task in an autonomous system**



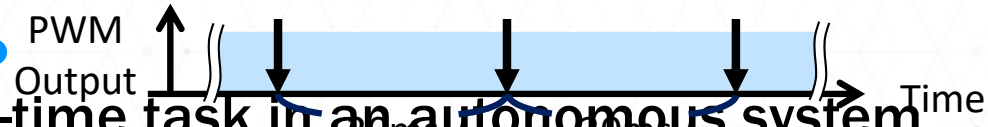
DEMONSTRATION 2

- Imagine attacker controls a real-time task in an autonomous system

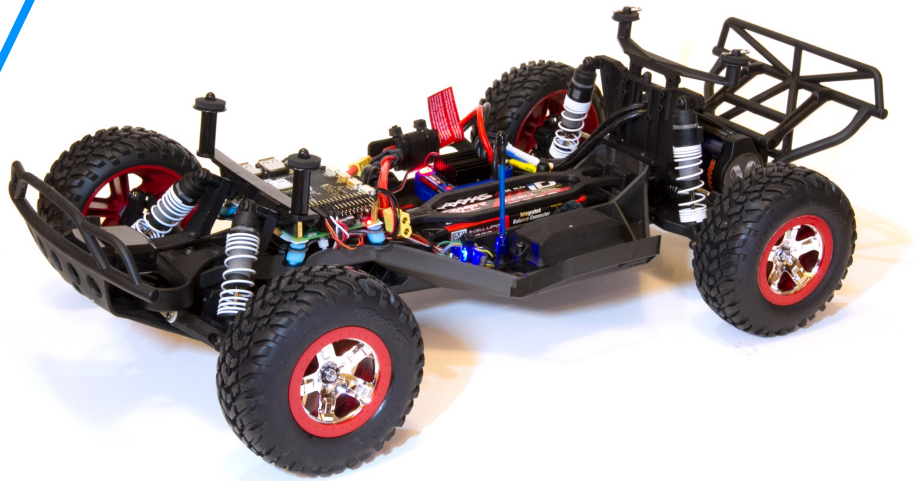


- It wants to take over control of the steering and throttle

DEMONSTRATION 2



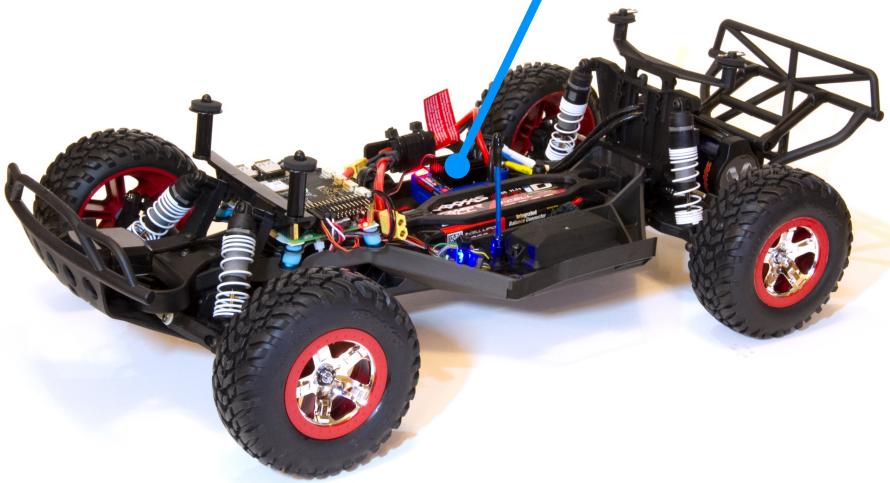
- Imagine attacker controls a real-time task in an autonomous system

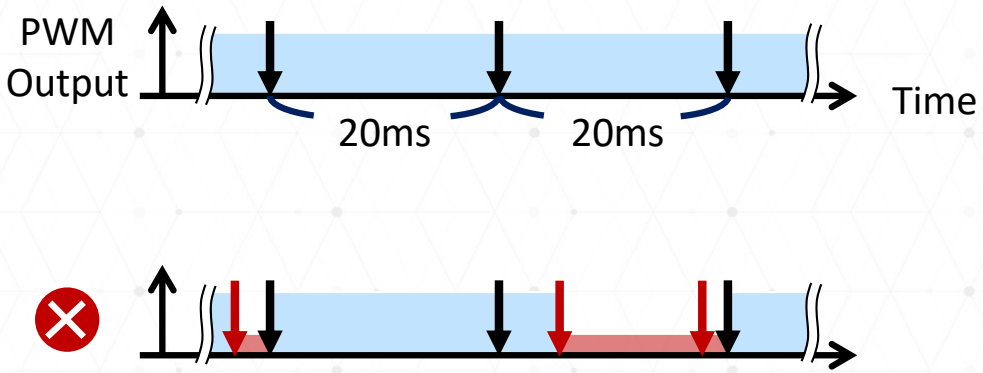


- It wants to take over control of the steering and throttle

PWM Update Task ↓ Attacker's Task ↓

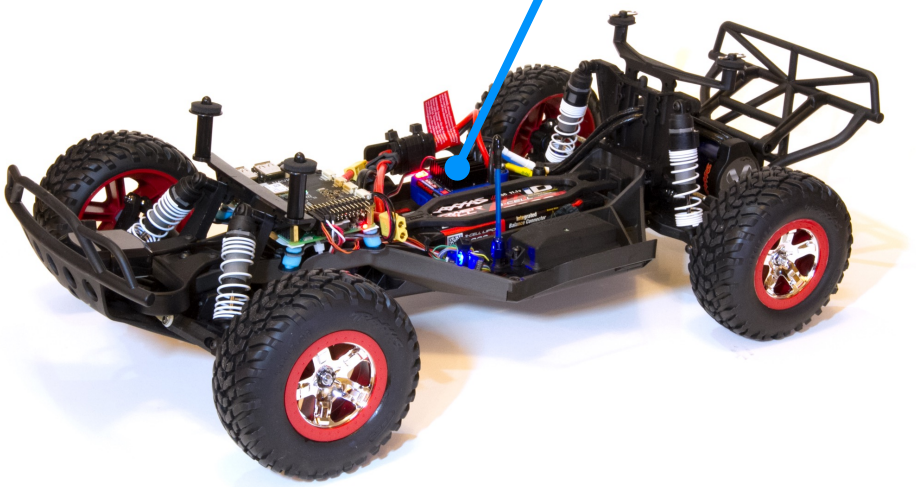
DEMONSTRATION 2

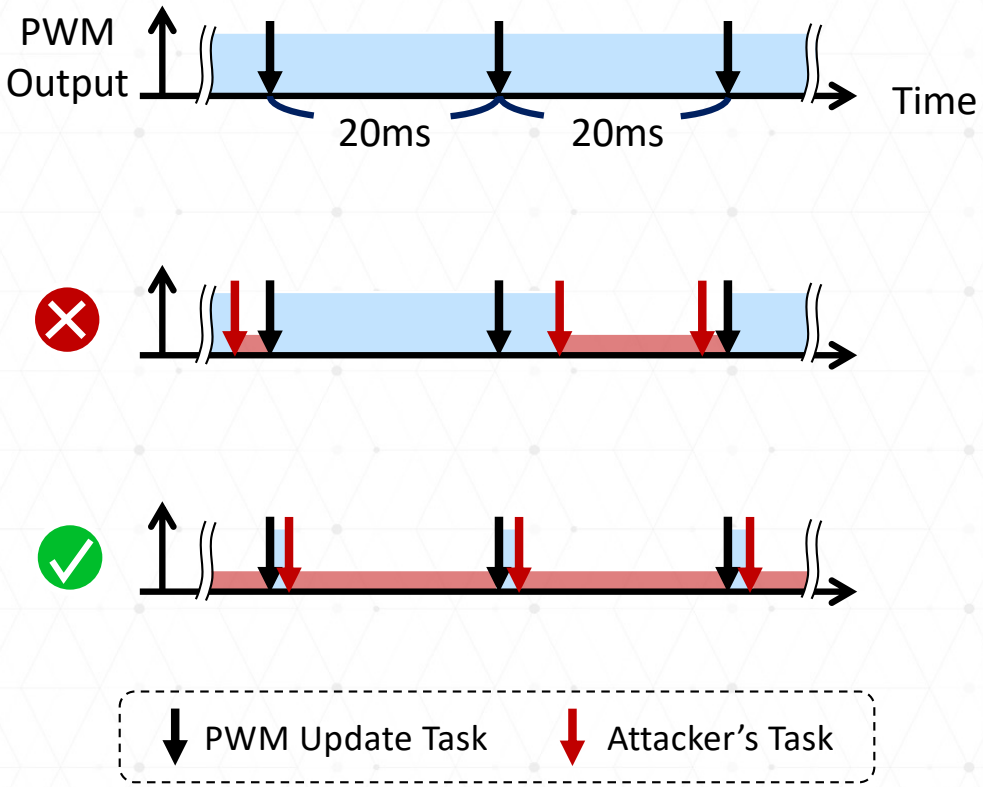


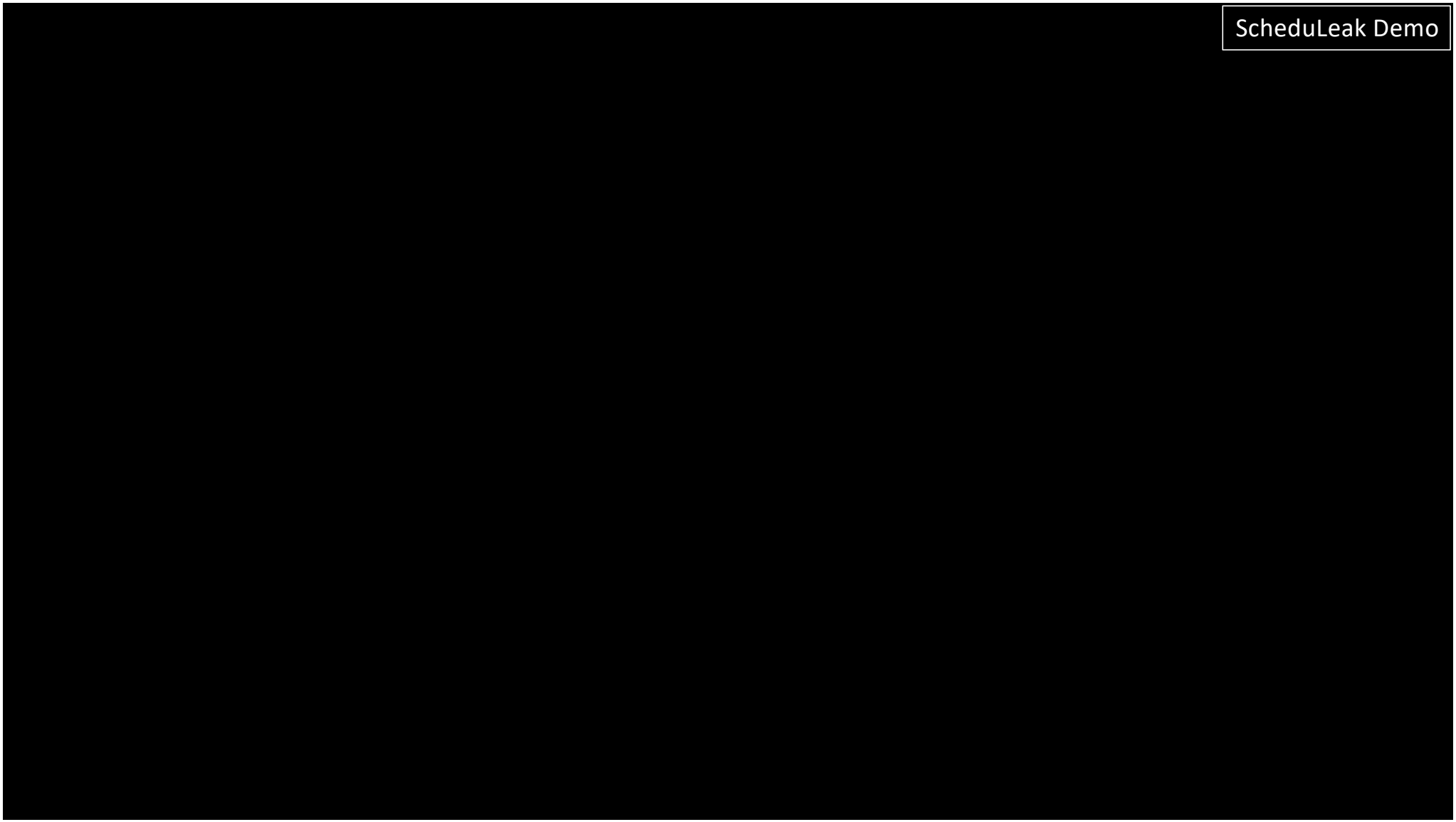


PWM Update Task Attacker's Task

DEMONSTRATION 2

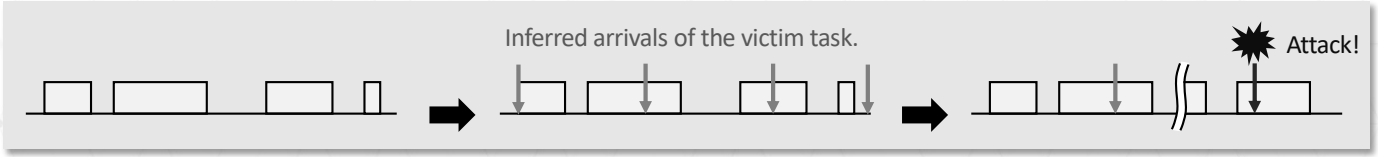





ScheduLeak Demo

SCHEDULEAK SUMMARY

- Reconnaissance attack algorithms
- Targeting sporadic and mixed real-time CPS
- Stealthy and Effective
- **No root privileges** required for ScheduLeak



More videos [including cache attack demo]: **https://scheduleak.github.io**