Machine Learning Enhanced Real-Time Intrusion Detection Using Timing Information

Hang Xu, Frank Mueller

Mithun Acharya, Alok Kucheria

Outline

Motivation

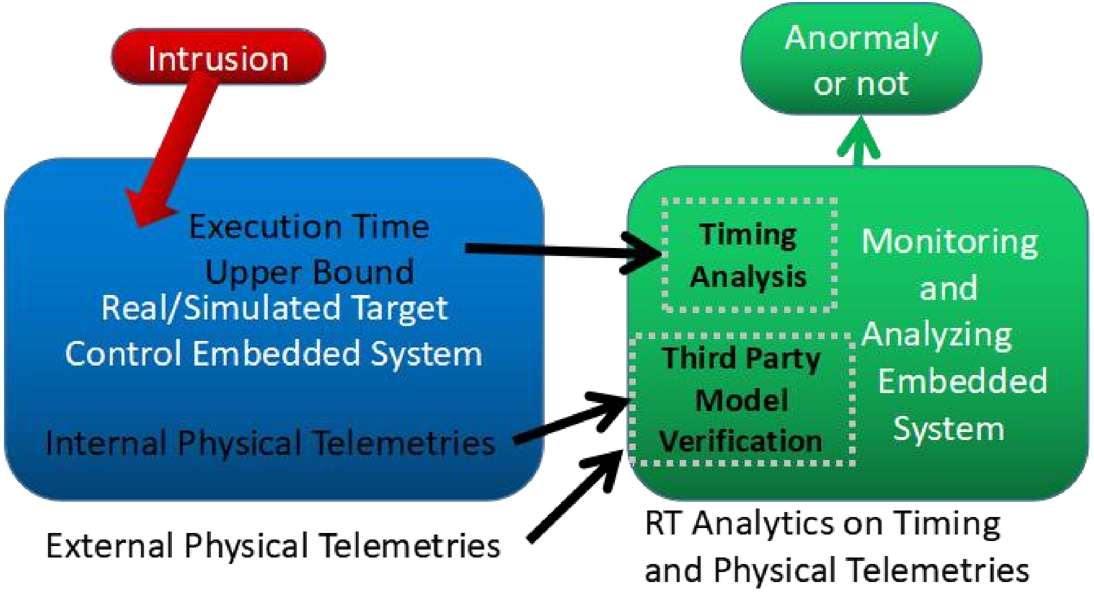
- 1. Rising cyber attacks toward Internet-of-Thing(IoT) systems / embedded systems
- 2. Insufficient traditional intrusion detection methods
- 3. Migration of ML from cloud to edge

Past Work

1. Timing analysis
2. Physical model verification
3. Packet encryption
4. Communication frequency

Our Work

1. Separate intrusion detection system and target system
2. ML model for state verification
3. Real-time suitability optimization of ML library



Outline

1. Motivation
2. Detection Analyses
3. RT Performance of ML Library
4. Application and Experiments
5. Future Work and Ackownledgement

Two Analyses

- Timing Analysis
  - Execution time of certain paths in program code
  - Communication delay between the detector and controller (Timestamps of communication in packets under monitoring)
- Third Party Model Verification
  - Internal Physical Telemetries
  - External Physical Telemetries

Detection Algorithm

Packet arrival timeliness validity (detector blockingly wait; exclude packet buffering delay)

Execution time validity

ML model inference comparison

function DETECT_ANOMALY(*socket*)
    $Data = read(socket);$
    $\boxed{T_{dtc} = gettimeofday();}$
    if $Data \leq 0$ **then**
        return True; ▷ packet not received
    else
        $[T_{tgt}, T_{ctrl}[N], PHY_{in}, PHY_{ex}] = parse(Data);$
        if $T_{dtc} - T_{tgt} > D_{comm}$ **then**
            return True; ▷ data packet not received in time
        else if $\exists i, T_{ctrl}[i] > WCET[i], 0 \leq i < N, i \in Z$ **then**
            return True; ▷ execution time over bound
        else
            $[MSR_{in}, MSR_{out}] = select\_telemetry(PHY_{in}, PHY_{ex})$
            $EXP_{out} = ML\_Model(MSR_{in})$
            if $\|EXP_{out} - MSR_{out}\| > TH_{ML}$ **then**
                return True; ▷ ML verification failed
            else
                return False; ▷ No anomaly
            end if
        end if
    end if
end function

PARAMETERS OF THE DETECTION ALGORITHM

Symbol	*Description*
N	vector size of execution time for different code snippets on the target control code
D_{comm}	deadline of communication delay
$WCET[N]$	worst-case execution time vector
TH_{ML}	ML model verification threshold
socket	controller socket file descriptor
Data	streaming data from controller
T_{dtc}	the data reception timestamp on the detector
T_{tgt}	the transmission timestamp on the controller
$T_{ctrl}[N]$	execution time vector on target control system
PHY_{in}	internal physical telemetries
PHY_{ex}	external physical telemetries
MSR_{in}	physical telemetries selected as measured inputs
MSR_{out}	physical telemetries selected as measured outputs
EXP_{out}	inference output of ML model

Fusion of Analyses

- Assign weights for thresholds of different detection metrics
- Fuse to obtain entire detection threshold
- Counter stealthy attack

Increased Isolation

1. Between application space and OS kernel space (execution time information managed by kernel)

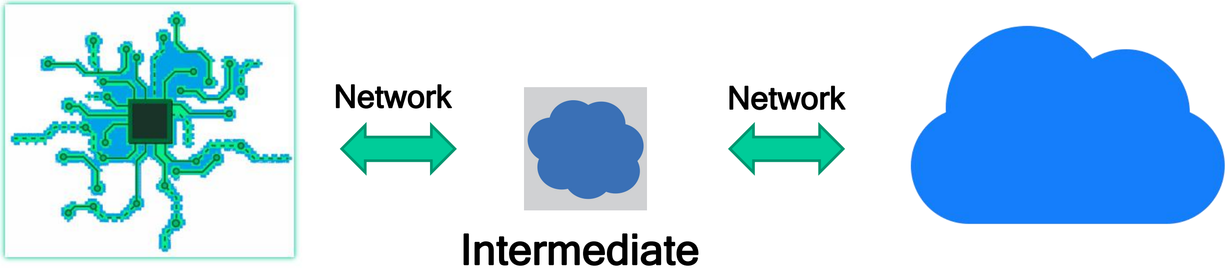
2. Between controller and detector

3. Between internal and external physical state data sources

Outline

1. Motivation
2. Detection Analyses
3. RT Performance of ML Library
4. Application and Experiments
5. Future Work and Ackownledgement

Edge Computing



Edge Computing

Intermediate Service

Cloud Computing

Why does ML move?

- **Real-Time features of ML API on edge**
  - Shorter average execution time
  - Tighter worst case execution time (WCET)

- Large streaming data inputs
- Data privacy concerns
- Lower latencies

Real-time Optimization for ML Library

- 1. Insufficient real-time predictability of keras and original Caffe libraries
- 2. Reducible detection delay

  - promptness of detection

  - compatibility with high sampling rate of control system

ML Libraries

1. Keras (Tensorflow backended)
   - Interpreter-based language
   - No real-time control of dynamic memory management
2. Caffe
   - Native C/C++ language
   - Real-time control of dynamic memory management
3. Enhanced Caffe
   - Remove third party library invocation functions in source code
   - Remove multi-core support

RT Performance Comparison

Keras vs. Original Caffe

Average execution time

- 4:1

Stadard deviation of execution time

- less varying : much more varying

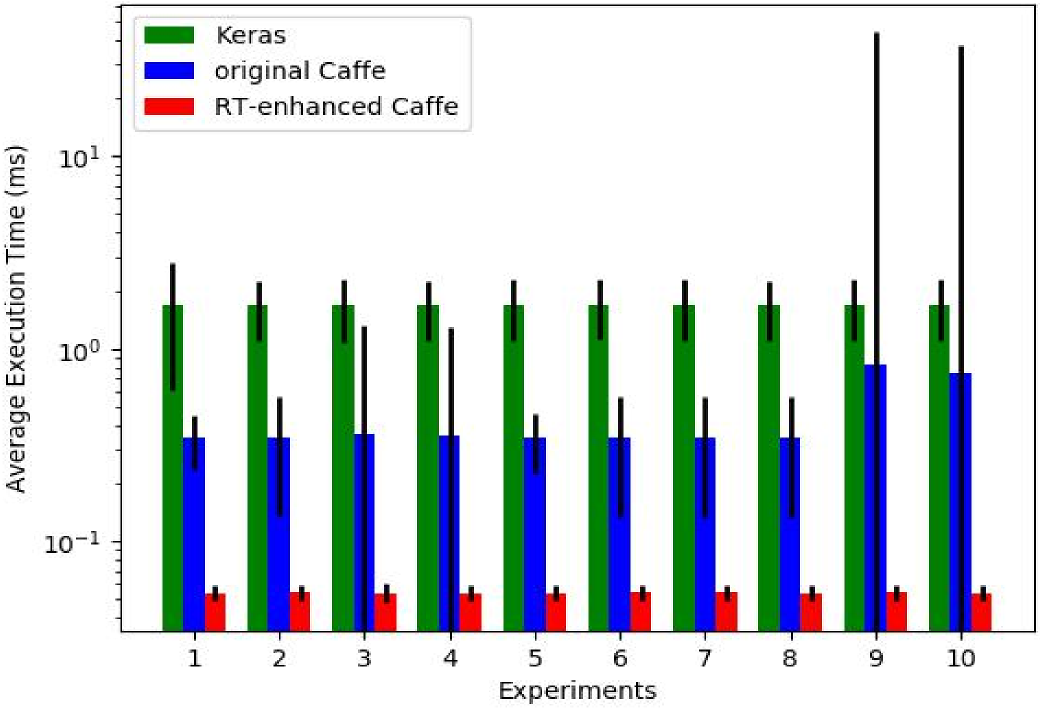
RT-Enhanced Caffe vs. original Caffe

Average execution time

- 1: 6

Standard deviation of execution time

- 1:25 (comparison between the minimum values)



Outline

Application for Intrusion Detection

- 1. Solar power system
- 2. Power output estimation
- 3. Inverter controlled by DSP (embedded system)
- 4. Inverter is emulated by Pi3 B

Model Preparation

- Model training
  - on HPC system with GPU support
  - Keras tensorflow backended
  - model converted into caffe compatible format
- Model inference
  - on raspberry Pi

RT predictability; embedded system; power efficiency; ML infrastructure support

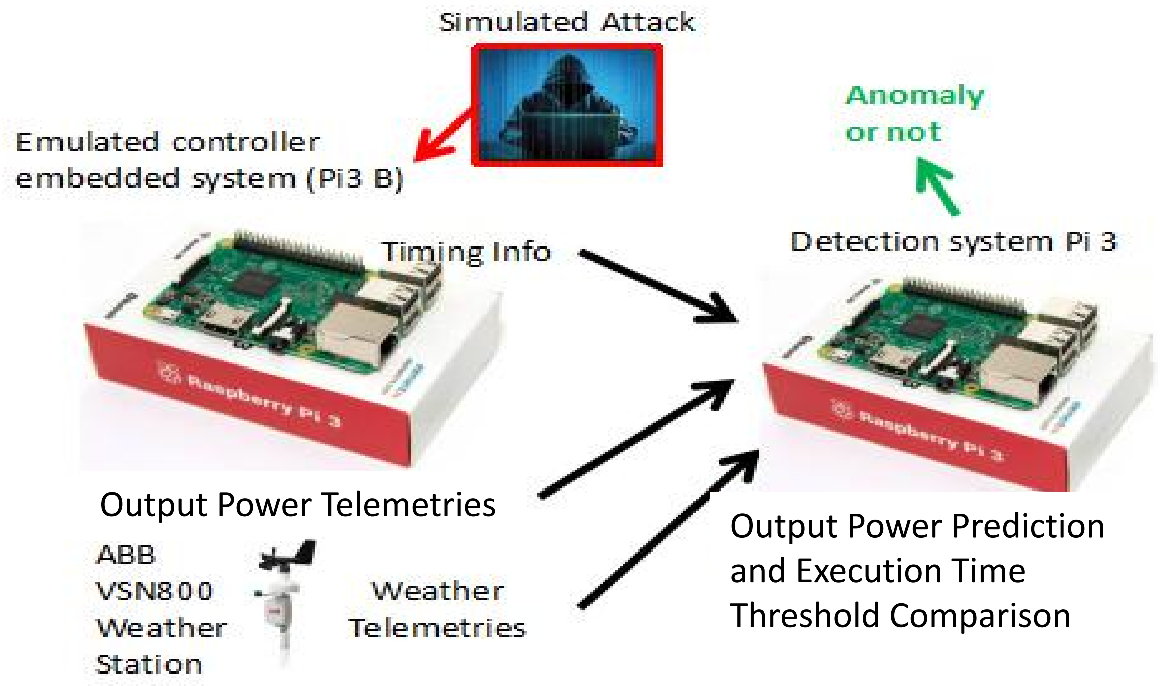
Experiment of Intrusion Detection System

Verification model:

1. Use ANN.
2. Use RT-enhanced Caffe library.
3. Training based on practical industrial data from ABB
4. Mode accuracy: 0.891 (explained variance score)

Simulated Attack:

Simulated execution time variation by sleeping time



Experiment Configuration

- 2kw maximum model output magnitude; percentage of prediction error threshold: 1%, 5%, 10%.
- 10ms valid execution time; percentage of execution time deviation threshold: 0.1%,1%,10%
- Communication delay upper bounded 2.7ms
- 2500 samples of timing and ML prediction each experiment run
- 80% samples with intrusion(20% samples with timing deviation less than the smallest timing threshold), 20% without intrusion

Results

1. Inverter ouptout power deviation threshold larger,

    FP smaller

    FN larger

    accuracy decreases

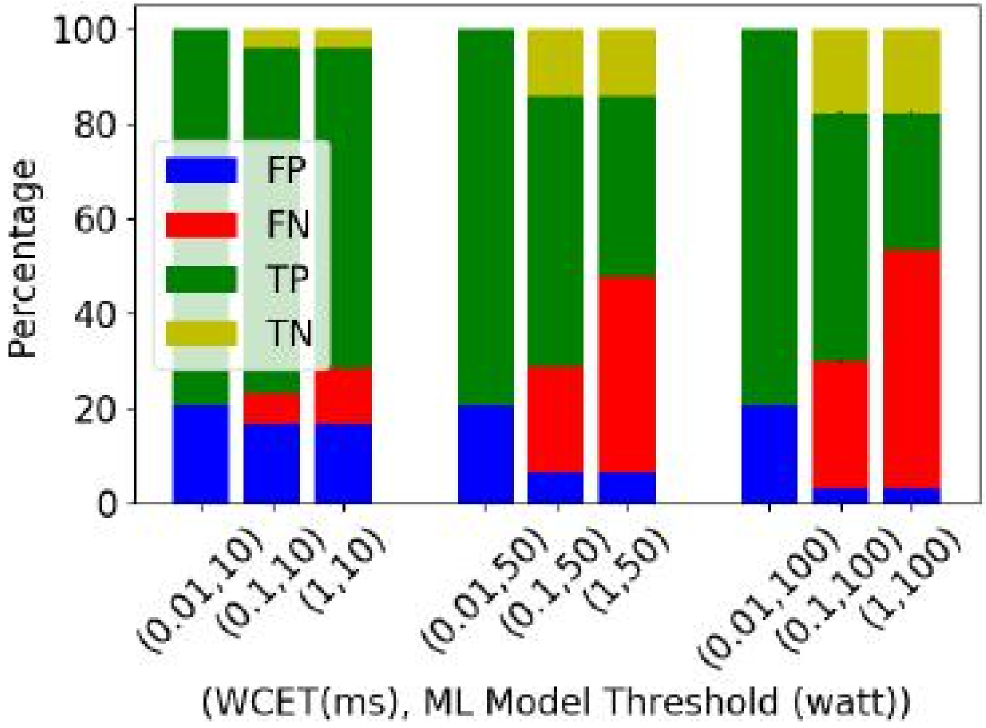
2. Timing threshold larger,

    FP smaller

    FN larger

    accuracy decreases

3. False positive and false negative rate vary in opposite directions



Conclusion

1. We enhanced prior intrusion detection based on timing analysis via ML model verification.

2. We conducted experiments to demonstrate its effectiveness based on practical industrial data.

3. We investigated the trade-off between FP and FN rates when selecting the detection thresholds of WCET and ML model output.

Outline

1. Motivation
2. Detection Analyses
3. RT Performance of ML Library
4. Application and Experiments
5. Future Work and Ackownledgement

Future Work

- Timing analysis based cyber protection
  - Network stack
  - User library for open source programs
  - Runtime library
  - OS Kernel
  - Parametric timing analysis
  - Safe-mode transition

Acknowledgement

This work was funded in part by NSF grants 1329780, 1525609, and 1813004.

Thank you!