Timing Based Security in Real-Time Systems: T-SYS & T-Pack

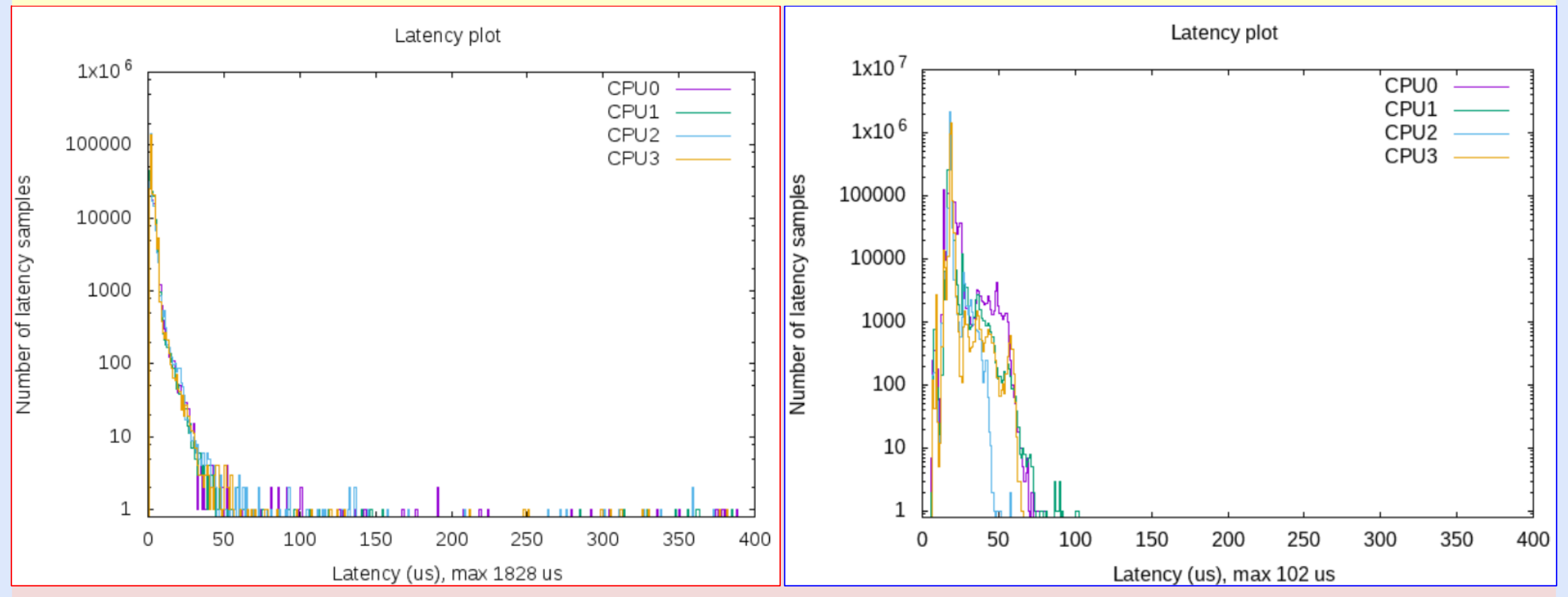
Brayden McDonald & Swastik Mittal

North Carolina State University
Advisor: Dr. Frank Mueller

Motivation

- Real-time systems
  - Timeliness considered part of correctness
  - Predictability over performance
  - Execution time analysis is part of development
  - Common example: cyberphysical systems
- Attacks against real time systems
  - Increasing threat
    - Computers more ubiquitous than ever
  - Attackers can subvert or damage critical infrastructure
- Susceptible to delay attacks
  - Delay intended execution time of real-time system
  - Network attacks
    - Denial of service attack leads to network delay

Background



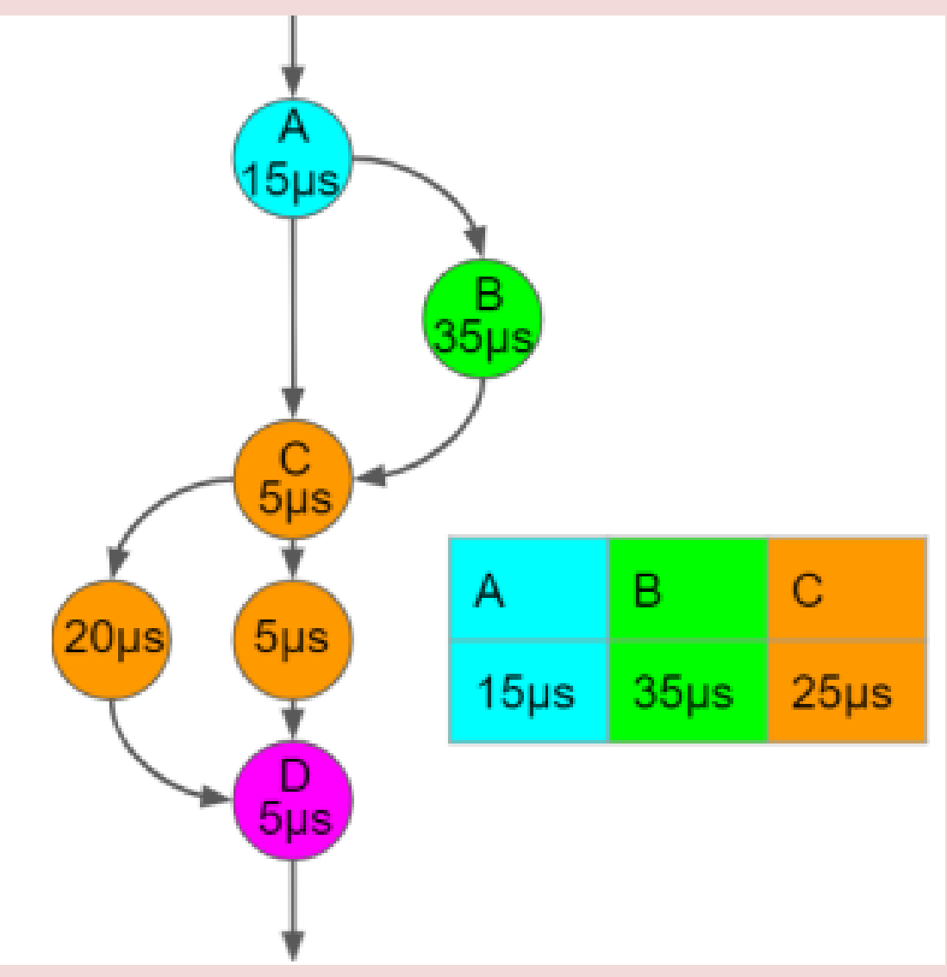
- Latency results
  - Linux vs Preempt-RT Linux (Raspberry Pi 3)
- Latency - Difference between thread wake up call vs actual thread wake up time
- Preempt-RT Linux: higher average latency, but **no outliers**

Solution

- T-Sys
  - Intrusion detection in real-time systems using timing anomalies
    - Automatic compiler-based integration
    - Configure protection/performance tradeoff using user-defined *maxvuln* parameter
  - Able to detect 100% of attacks where duration exceeds *maxvuln*
- T-Pack
  - Timed network security framework to detect intrusion on the network
    - Intrusions leading to unwanted delay of useful packets.
  - Able to detect 100% of the DDOS attack of minimal intensity with a minimum cost overhead.

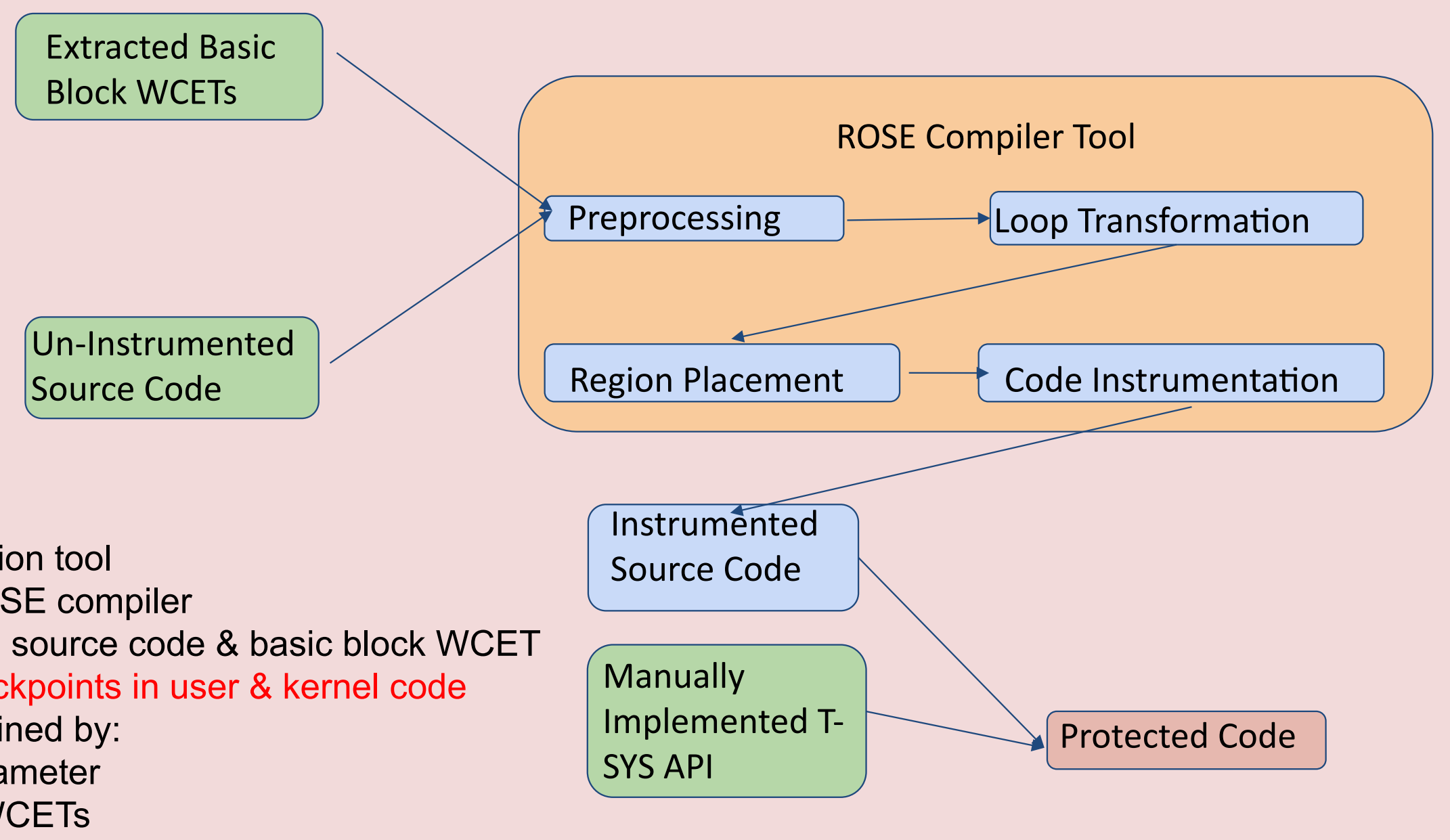
T-SYS

- Insert instrumentation points into target codebase
  - track progress through a known execution path
- Set up deadline at each point
  - Must reach next point by deadline
- If deadline is missed, assume intrusion
  - system goes to safe mode
- Maximum allowed deadline value called MaxVuln
  - MaxVuln set by user



A	B	C
15μs	35μs	25μs

T-SYS



Extracted Basic Block WCETs

Un-Instrumented Source Code

ROSE Compiler Tool

Preprocessing

Loop Transformation

Region Placement

Code Instrumentation

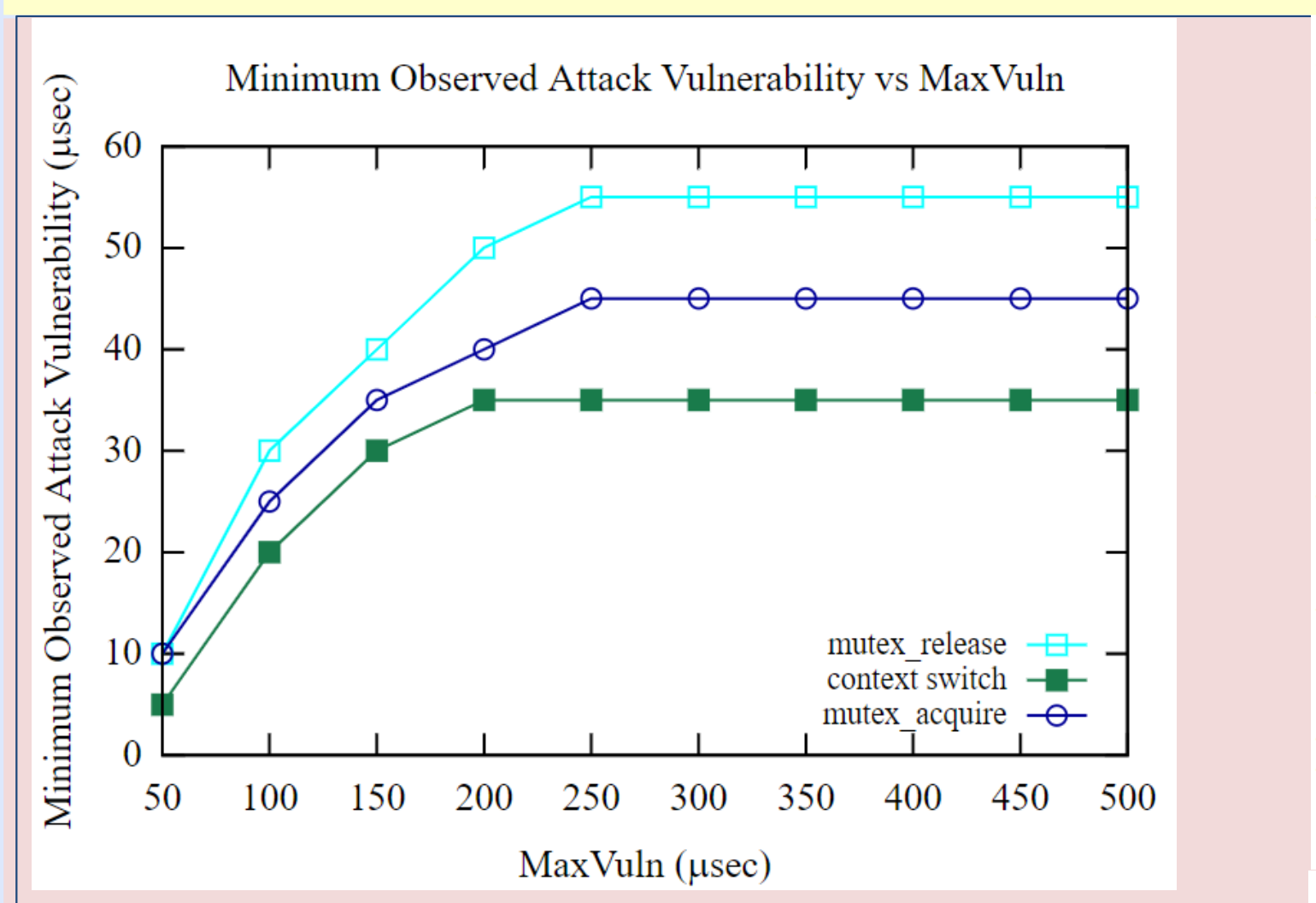
Instrumented Source Code

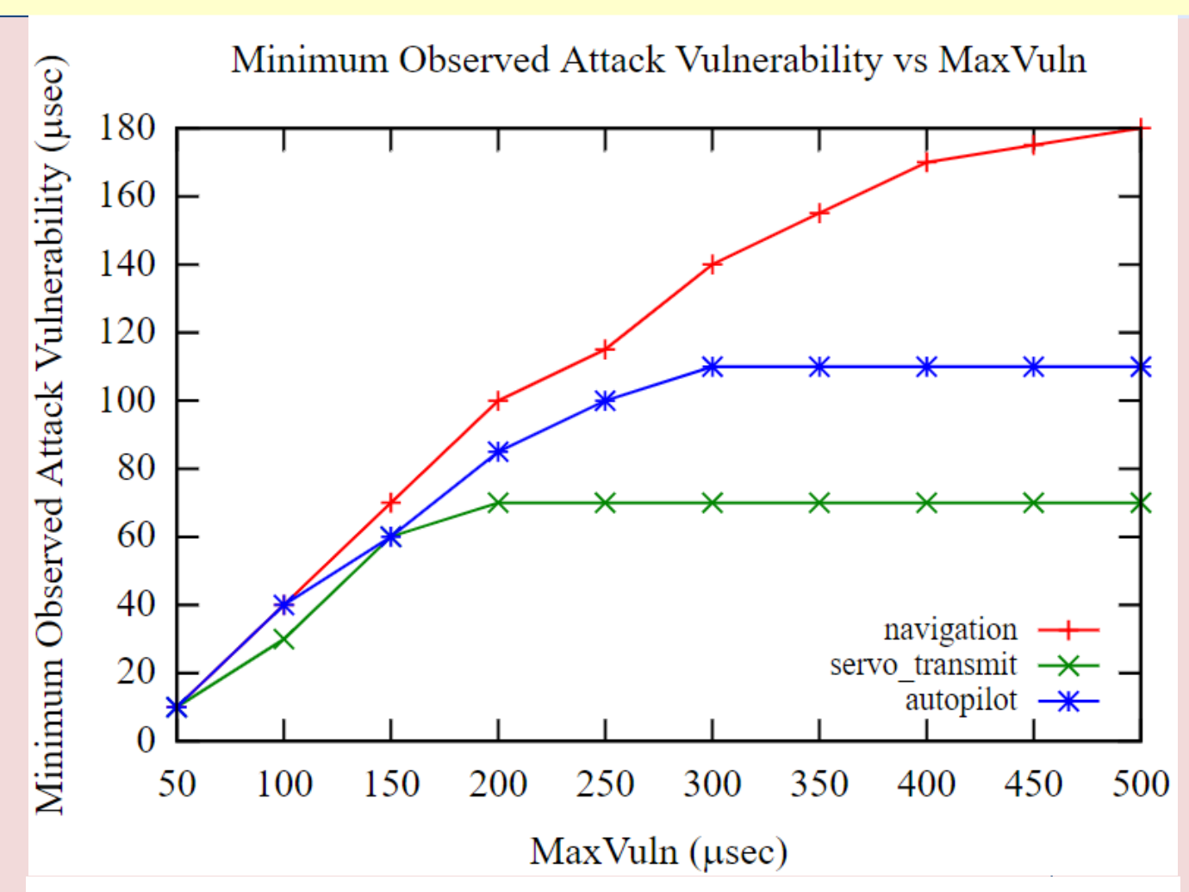
Manually Implemented T-SYS API

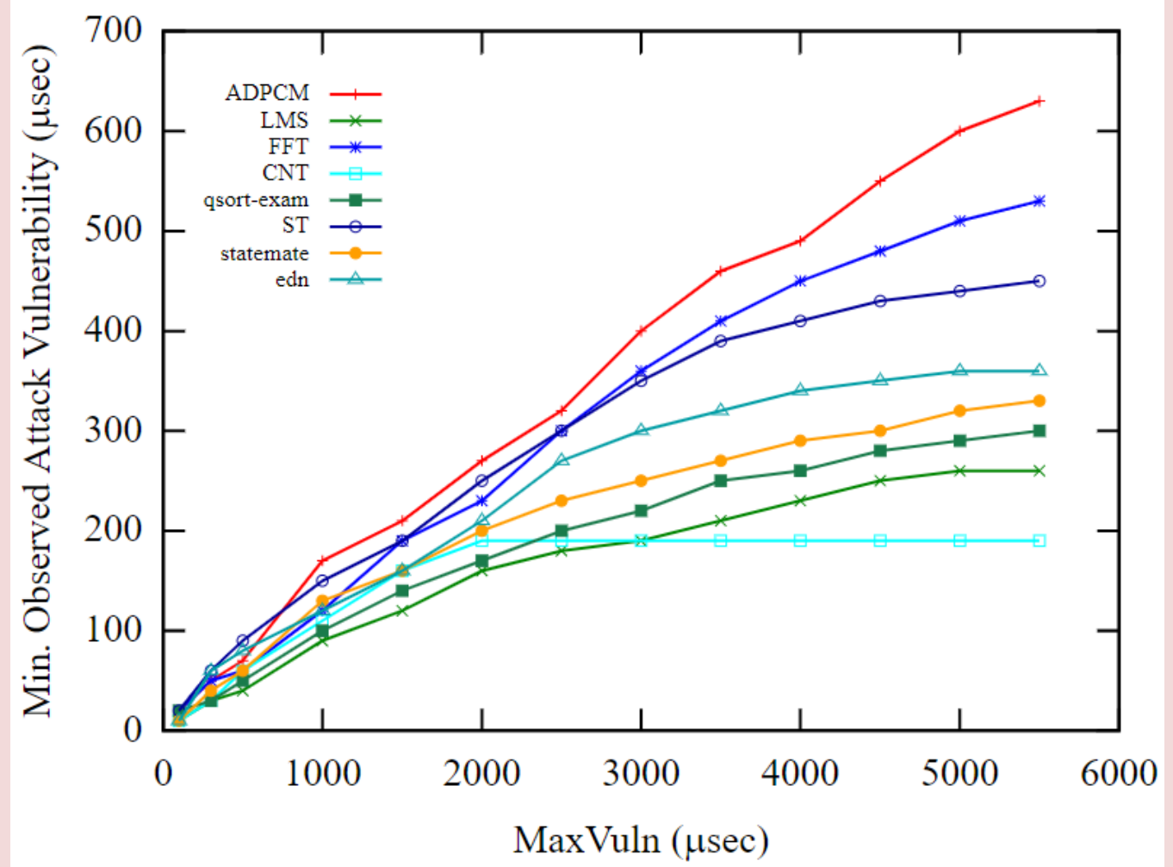
Protected Code

- Automatic integration tool
  - Based on ROSE compiler
  - User supplies source code & basic block WCET
- Can integrate checkpoints in user & kernel code
- IP position determined by:
  - *maxvuln* parameter
  - basic block WCETs

T-SYS Results







- Simulated attacks conducted using various MaxVuln values
  - Function calls inserted after instrumentation
  - Placed immediately after IP
    - optimal position for attacker
- Benchmarks used
  - Malardalen
  - modified PapaBench tasks
  - selected kernel paths
- Increase in MaxVuln leads to an increase in Minimum observed attack vulnerability in all cases
- Minimum Observed Attack Vuln. stops increasing once MaxVuln > Task WCET

$MaxVuln$ (μsec)	unprotected	1000	2000	3000	4000	5000
adpcm	321079	613574	484969	458921	423463	362904
lms	518362	989697	782991	741223	684509	585666
fft	68315	130266	103367	97615	90156	76695
cnt	1981	2601	2226	1991	1992	1990
statemate	295433	563840	446305	422211	390409	334027
edn	147086	280464	221775	209940	193686	166191
qsort-exam	6518	12180	9848	9659	8603	6871
st	426710	813607	642774	609665	562184	481264

Table 2: Average execution time (in μsec) of Malardalen benchmarks for different values of $MaxVuln$.

$MaxVuln$ (μsec)	unprotected	100	200	300	400	500
navigation	614	1162	921	863	821	685
servo_transmit	186	262	199	201	197	198
autopilot	292	426	385	342	301	305
context_switch	157	197	176	157	158	156
mutex_acquire	245	297	278	246	244	245
mutex_release	221	271	245	223	221	220

Table 1: Average execution time (in μsec) of PapaBench tasks and kernel paths for different values of $MaxVuln$.

- Execution time overhead is of particular importance for real-time security.
- Overhead data gathered experimentally
- Performance improves as MaxVuln increases
  - plateaus if entire task within one region

- We compare T-SYS to a state-of-the-art timing-based security system (Bellec):
  - Total regions created
  - Total regions entered during execution
- Bellec algorithm is not elastic
  - only one MaxVuln value
  - Comparison uses multiples of this value
- T-SYS creates fewer regions
- T-SYS enters regions less frequently during execution

Task	Base MAW	Bellec	T-SYS	T-SYS (0.5x)	T-SYS (2x)	T-SYS (5x)
adpcm	9007	36	31	74	23	6
lms	1210	47	34	68	17	11
fft	1117	41	38	72	19	12
cnt	274	15	9	17	5	2
statemate	2970	21	19	34	13	7
edn	3155	32	26	49	18	10
qsort-exam	614	25	23	62	14	9
st	8001	18	16	28	9	5
navigation	121	5	5	9	3	1
servo_transmit	93	3	3	5	1	1
autopilot	134	7	6	10	4	1

Table 3: Comparison of Bellec vs T-SYS algorithms, by number of regions created.

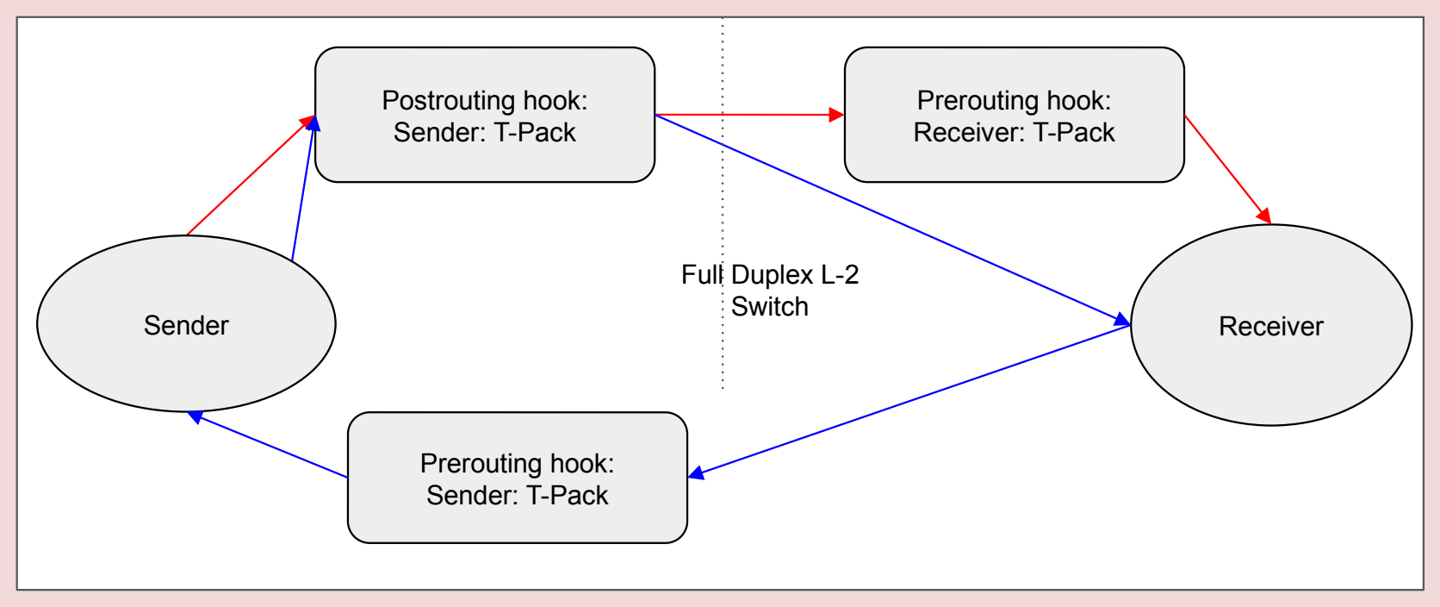
Task	Base MAW	Bellec	T-SYS	T-SYS (0.5x)	T-SYS (2x)	T-SYS (5x)
adpcm	9007	14256	12275	24912	6240	1504
lms	1210	407	351	906	241	191
fft	1117	2017	1736	3302	960	580
cnt	274	534	498	1011	278	101
statemate	2970	791	754	1294	452	239
edn	3155	1125	1052	1926	618	348
qsort-exam	614	971	956	1835	572	320
st	8001	640	601	1209	384	198
navigation	121	521	513	1017	221	71
servo_transmit	93	312	254	531	61	61
autopilot	134	548	457	1102	246	87

Table 4: Comparison of Bellec vs T-SYS algorithms, by number of regions entered during execution.

Nicolas Bellec, Simon Rokicki, and Isabelle Puaut. 2020. Attack detection through monitoring of timing deviations in embedded real-time systems. In ECRTS 2020 - 32nd Euromicro Conference on Real-Time Systems. Modena, Italy, 1–22.

T-Pack

- Linux netfilter hooks execute T-Pack module
- UDP packets between sender and receiver in red.
  - Add timestamp information to each packet
- TCP between sender and receiver in blue.
  - Initiate a timeout for each sent packet
    - maintain record in a Queue



Postrouting hook: Sender: T-Pack

Prerouting hook: Receiver: T-Pack

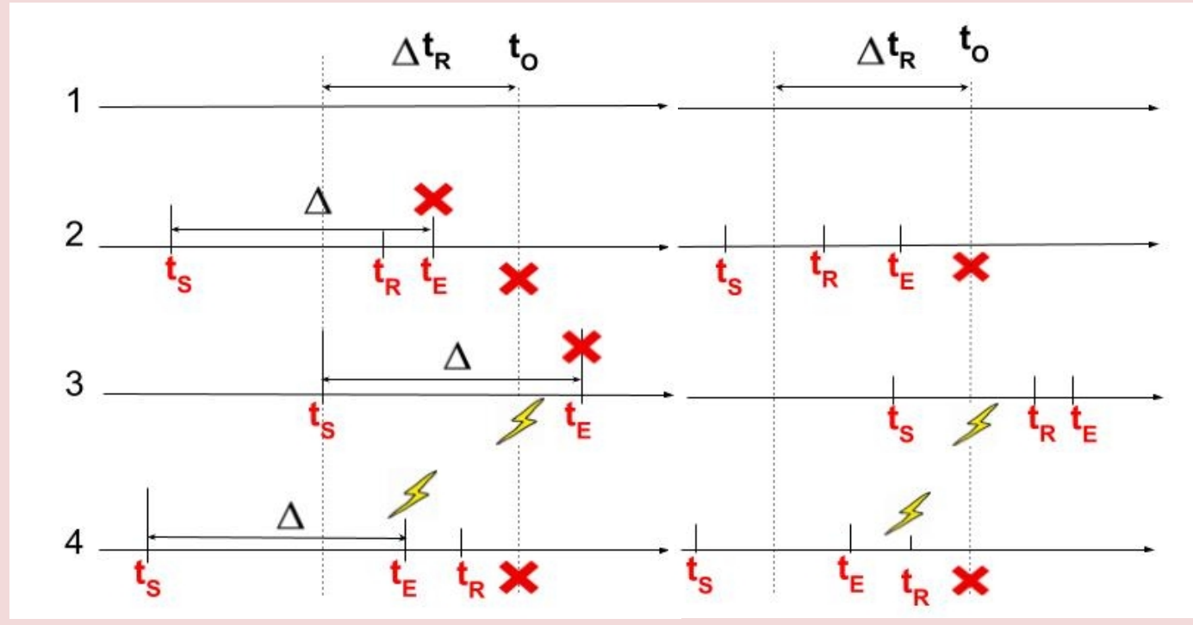
Sender

Full Duplex L-2 Switch

Receiver

Prerouting hook: Sender: T-Pack

T-Pack Implementation



a) TCP b) UDP

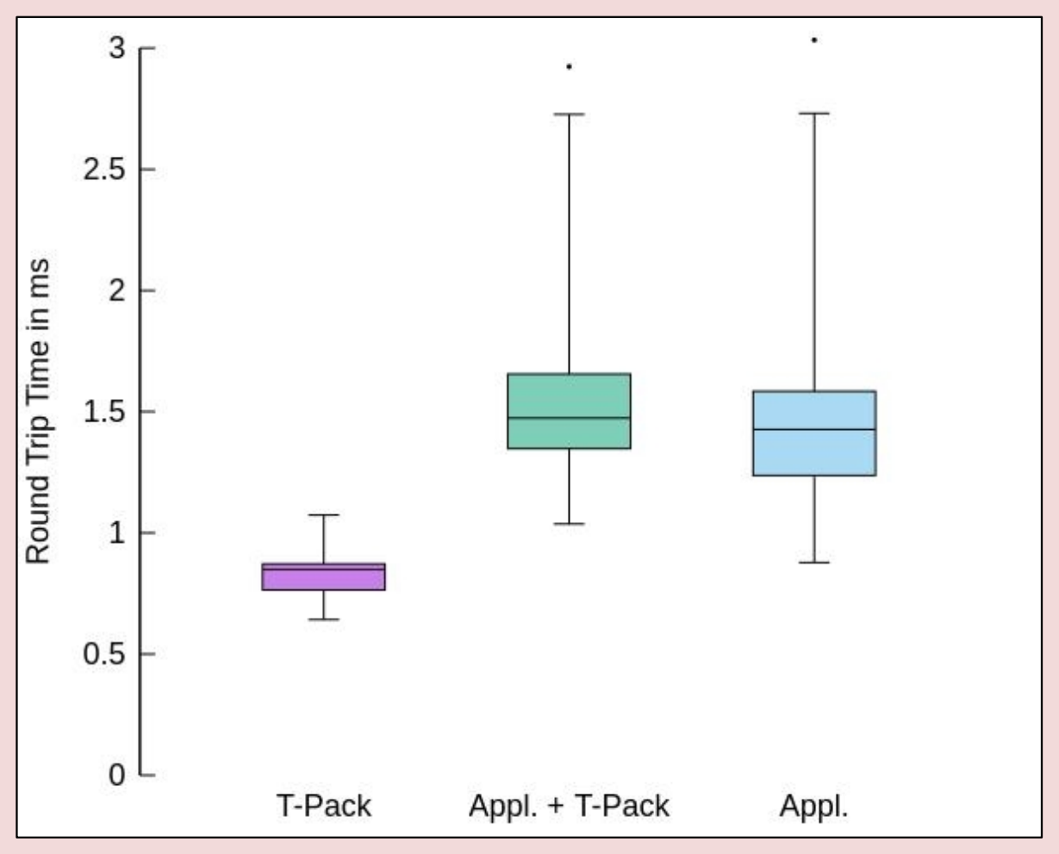
T-Pack Compatibility and Comparison to Global Timeout
t_S(Send Time), t_R (Receive Time) & t_E (Expected Receive Time)

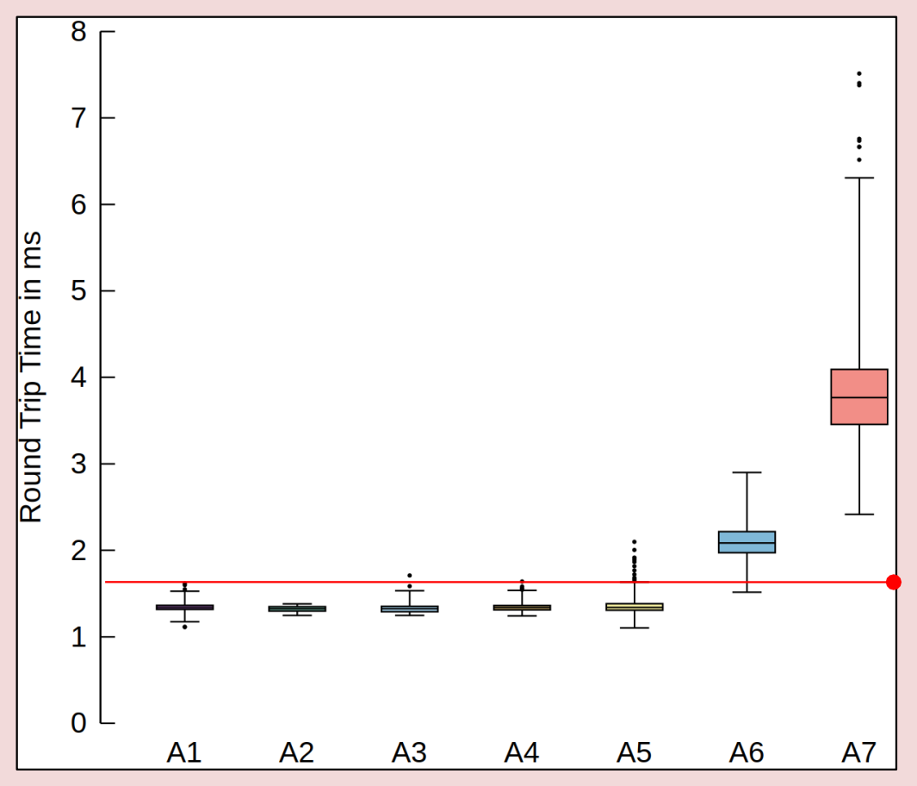
1. **T-Pack works with global timeouts**
2. In time arrival of packet at t_R
   - Cancelling t_E and t_O
3. Long delay before packet sent or a lost packet
   - t_O Cancel t_E
   - T-Pack needs global timeouts
4. Packet sent early or Late arrival of packet
   - Early timeout at t_E leaving t_O-t_E to transition into safe mode.
   - **Early intrusion detection with T-Pack**

- UDP: Exception raised at t_R for late arrival of a packet
  - Receiver unaware of the sender
  - t_R-t_O time to transition to safe mode

T-Pack Results

- Round-trip time (RTT) measured between two nodes in UAV Paparazzi model
  - Time between packet sent to ack received (TCP)
- RTT values measured by T-Pack at network layer
  - Early detection of delay attack
    - Eliminates transition time between layers of network stack
  - Cost effective
    - No hardware support needed
- RTT measured at Application layer with T-Pack and without T-Pack
  - Minimal effect on average performance (0.09ms)
- **T-Pack vs Baseline (appl. layer)**
  - Explicit reply packets needed to replicate Acks
    - saturating write buffer
    - Higher expected RTT →higher false negatives





Result: No attack A1:P(0,0,0,0), attacks A2:P(1,10,500,0.5), A3:P(1,10,500,0.1), A4:P(2,10,500,0.1), A5:P(2,30,500,0.05):, A6:P(2,10,500,0), A7:P(2,30,1000,0.001

- Result: RTT values of packets between two nodes of a Paparazzi UAV model
  - Distributed denial of service attack of varied intensity applied
  - Packet protected with IPSec encryption
    - T-Pack compatible with other security protocols
- **Attack Vector P(n,t,b,i) (ping of death)**
  - n attack nodes
  - t parallel threads each
  - b bytes of attack packet
  - i seconds time interval
- 100% of delay attacks resulting in min RTT above red line detected
  - Worst case RTT for no attack (A1)
- Vulnerability of T-Pack
  - Delay attacks with minimal intensity not always detected
  - A2 not detected by T-Pack

Publications

- T-SYS: Timed-Based System Security for Real-Time Kernels, by B. McDonald and F. Mueller International Conference on Cyber-Physical Systems (ICCPS), May 2022
- T-Pack: Timed Network Security for Real Time Systems Swastik Mittal, Frank Mueller in IEEE International Symposium on Real-Time Computing (ISORC), May 2021
- CLAIRE: Enabling Continual Learning for Real-time Autonomous Driving with a Dual-head Architecture Hao Zhang, Frank Mueller in IEEE international Symposium on Real-Time Computing (ISORC), May 2022

Acknowledgements

This work was supported in part by NSF grants 1813004 and 1525609

The 5th NSF Secure and Trustworthy Cyberspace Principal Investigator Meeting
June 1-2, 2022 | Arlington, Virginia