# CPS-MT: A Real-Time Cyber-Physical System Monitoring Tool for Security Research

Martín Barrère*, Chris Hankin*, Angelo Barboni*, Giulio Zizzo*, Francesca Boem§, Sergio Maffeis*, Thomas Parisini*
* Imperial College London, UK
{m.barrere, c.hankin, a.barboni16, g.zizzo17, maffeis, t.parisini}@imperial.ac.uk
§ University College London, UK
{f.boem}@ucl.ac.uk

***Abstract*—Monitoring systems are essential to understand and control the behaviour of systems and networks. Cyber-physical systems (CPS) are particularly delicate under that perspective since they involve real-time constraints and physical phenomena that are not usually considered in common IT solutions. Therefore, there is a need for publicly available monitoring tools able to contemplate these aspects. In this poster/demo, we present our initiative, called CPS-MT, towards a versatile, real-time CPS monitoring tool, with a particular focus on security research. We first present its architecture and main components, followed by a MiniCPS-based case study. We also describe a performance analysis and preliminary results. During the demo, we will discuss CPS-MT's capabilities and limitations for security applications.**

## I. Introduction

In recent years, cyber-physical systems (CPS) have attracted a lot of research attention due to their intrinsic combination of cyber and physical aspects, in particular on security concerns [1]. One of the main challenges is therefore to rethink security from a new angle where cyber, physical and human aspects are integrated and understood as a whole. In addition, CPS systems are usually time critical and thus, real-time aspects are fundamental. In this work, we focus on industrial control systems (ICS), which are a class of cyber-physical system where cyber entities (e.g. computers, PLCs) monitor and control processes and physical actions. Cyber attacks on these systems may have serious physical consequences such as flooding, blackouts, or even nuclear disasters [1]. Hence, ICS security is vital since its compromise may result in a myriad of severe problems, from service disruptions and economic loss, to jeopardising natural ecosystems and human lives.

To the best of our knowledge, the security research community lacks public and configurable monitoring tools that can be adapted to different CPS research scenarios and consider real-time aspects as well. We often see monitoring solutions that query a server every five seconds to display updated values on screen. This is clearly not real time, involves unnecessary computation load, and may also crash the client. In addition, CPS environments present widely diverse configurations across different domains and custom setups. CPS-MT aims at covering these needs by means of a versatile, real-time oriented architecture, and a simple, yet expressive, configuration mechanism. A demo of CPS-MT is available online at [2]. We plan to release CPS-MT as an open-source project soon.

This work is supported by the KIOS Research and Innovation Center of Excellence, European-founded initiative (H2020, ID 739551).

## II. Monitoring approach

Our monitoring approach relies on the architecture illustrated in Fig. 1. The upper layer represents the CPS elements to be monitored, e.g., readings from PLCs, actions, sensor readings, etc. The middle layer acts as a broker between the elements being monitored and CPS-MT. We use Redis [3] to implement this layer. Redis is a fast in-memory database that stores data in the form of key-value pairs. The main idea is that monitored elements publish their data via Redis channels and CPS-MT subscribes to these predefined channels in order to receive updates in real time. This makes CPS-MT almost agnostic to what is being monitored, and thus very flexible.

The bottom layer illustrates the main components of the CPS-MT client-server architecture. The main goal of the server is to monitor the activity on Redis channels and report updates to the client side. The client (Web browser) will display and/or capture this new data as it becomes available. We use WebSockets to implement a two-way communication between the server and the client [4]. This allows the server to push data directly into the client in real time. WebSockets eliminate long polling and multiple client requests as it happens with traditional HTTP-based approaches.

Visualisation aspects are handled by the client side and rely mostly on JavaScript, D3.js [5] and related technologies. Fig. 2 shows a snapshot of CPS-MT displaying three monitors in real time. The system also allows to explore captured sessions in order to analyse CPS behaviour over specific periods of time.

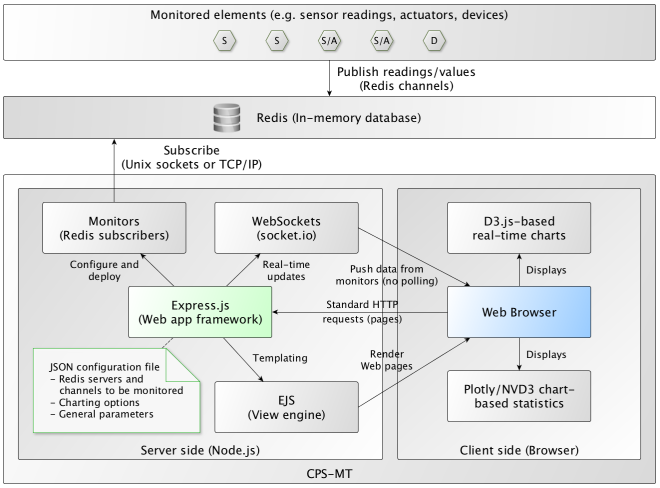


Fig. 1: CPS-MT high-level architecture


Fig. 2: CPS-MT real-time monitors [2]

## III. Case Study

This demo aims at showing and discussing CPS-MT's capabilities using auto-generated data as well as MiniCPS-based simulated environments. MiniCPS is an extensible Python-based simulation framework [6], built on top of Mininet [7], that implements simulated CPS components such as PLCs, their interactions with physical devices, and standard industrial protocols such as Modbus/TCP and CIP over Ethernet/IP. Our current research work focuses on ICS security, in particular on water treatment plants. MiniCPS provides a partial implementation of a real testbed developed at iTrust, Singapore [8], which we have used for our research work. MiniCPS uses SQLite to store the state of sensor readings and physical devices. We have extended MiniCPS to also support Redis as its data store, thus enabling CPS-MT to monitor the status of the whole simulation process, including cyber attacks. In that context, we will perform different kinds of attacks (e.g. man in the middle attacks), and will show how we can use CPS-MT to identify discrepancies and abnormal behaviour.

## IV. Preliminary Results

CPS-MT has shown a good performance on our case study. However, we have also conducted an intensive analysis to identify to what extent the client is able to properly display incoming events in real time and store session captures. Our methodology uses a 30-second window where simulated events are generated at different frequencies (in the order of milliseconds) and sent to the client in the form of stress tests. Observations are made until the client becomes unresponsive (or crashes). We also vary this analysis over a number of monitors ranging from one to ten, as shown in Fig. 3. We used a standard MacBook Pro (Mid 2014) for these experiments.

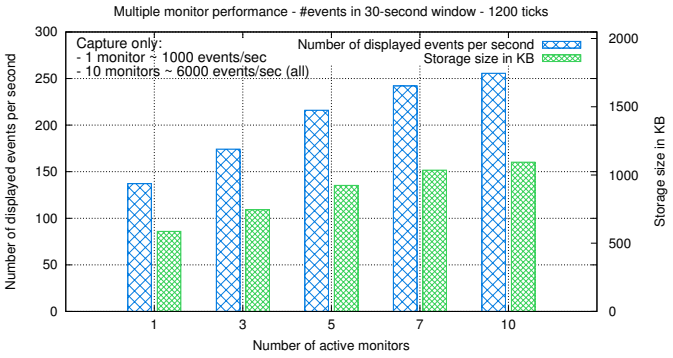


Fig. 3: CPS-MT performance analysis

In the case of one monitor, the system is able to properly display and capture up to 137.2 events per second, which corresponds to an average ingress rate of 7 milliseconds. This is more than enough for standard processes generating a few dozens events per second. The capture size is around 573.95 KB for a 30-second window. We have observed, however, that the overall manageable ingress rate is even higher when multiple monitors are considered. For three monitors, the system is able to properly display up to 174.3 events per second, while for ten monitors, it reaches 255.7 events per second across all monitors. This is due to the asynchronous nature of JavaScript which updates the charts independently.

Another goal of CPS-MT is the ability to capture data for specific sessions and generate datasets of real and/or simulated CPS events. This is particularly important for many research areas such as machine learning where datasets are essential. CPS-MT implements two ways of storing captured data: one uses the browser session storage (usually limited to 5-10 MB), and the other forwards received data to an external storage server via WebSockets. Our experimental results on capture only (no display) show that, for one monitor, CPS-MT is able to capture and send around 1000 events per second, while for ten monitors, the rate is close to 6000 events per second.

## V. Demo and Discussion

We will use these results to stimulate enriching discussions about state-of-the-art techniques and research issues yet to be solved, including noisy environments (e.g. jitter, delay), hardware in the loop capabilities, real-time visualisation aspects, scalability, and deployment of security countermeasures.

## References

[1] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-Physical Systems Security - A Survey," *IEEE Internet of Things*, vol. 4, pp. 1802–1831, Dec 2017.
[2] "CPS-MT - A Real-time CPS Monitoring Tool." http://demo.cpsmt.org/. Cited: June 2018.
[3] "Redis." https://redis.io/. Cited: June 2018.
[4] "RFC 6455 - The WebSocket Protocol." https://tools.ietf.org/html/rfc6455. Cited: June 2018.
[5] "D3.js - Data Driven Documents." https://d3js.org/. Cited: June 2018.
[6] D. Antonioli and N. O. Tippenhauer, "MiniCPS: A Toolkit for Security Research on CPS Networks," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, CPS-SPC'15, 2015.
[7] "Mininet." http://mininet.org/. Cited: June 2018.
[8] "iTrust: Secure Water Treatment." https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/. Cited: June 2018.