A Runtime Verification Framework For Cyber-physical Systems Based On Data Analytics And LTL Formula Learning

Ayodeji James Akande, Zhe Hou, Ernest Foo, and Qinyi Li

Griffith University, Australia
ayodeji.akande@griffithuni.edu.au,
{z.hou, e.foo, qinyi.li}@griffith.edu.au

Abstract. Safeguarding individuals and valuable resources from cyber threats stands as a paramount concern in the digital landscape, encompassing realms like cyber-physical systems and IoT systems. The safeguarding of cyber-physical systems (CPS) is particularly challenging given their intricate infrastructure, necessitating ongoing real-time analysis and swift responses to potential threats. Our proposition introduces a digital twin framework built upon runtime verification, effectively harnessing the capabilities of data analytics and the acquisition of Linear Temporal Logic (LTL) formulas. We demonstrate the efficacy of our approach through an application to water distribution systems.

Keywords: Runtime verification · Linear temporal logic · Digital twins · Formal Modeling · Cyber-physical systems · Cyber-security

1 Introduction

Safeguarding users and assets from cyber-attacks within the digital realm has grown into an essential concern. This realm encompasses components such as cyber-physical systems (CPS), the metaverse, satellite communication systems, and the Internet of Things (IoT), all of which hold substantial importance in both industrial operations and the intricate tapestry of human existence. This research explores the realm of enhancing the security of cyber-physical systems, the next generation of systems combining computational and physical capabilities, enabling interaction with humans through various new modalities [1]. In Cyber-physical systems (CPS), the physical and software components are closely intertwined, capable of operating on different spatial and temporal scales, demonstrating diverse behavioural modes, and interacting with each other in context-dependent ways [5].

Incidents targeting cyber-physical systems, encompassing domains like industrial automation, smart grids, smart cities, autonomous vehicles, and agricultural precision, have the potential to result in devastating outcomes for both individuals and assets. This paper centers on attacks targeting engineering and network data, emphasizing the need to confront and counteract this threat to guarantee the safety and security of cyber-physical systems.

Securing cyber-physical systems (CPS), due to the intricate nature of their infrastructure, requires continuous real-time analysis and swift action against potential attacks. Despite the proposal of methods such as intrusion detection/prevention systems and network mapping, these approaches frequently prove insufficient or encounter constraints in effectively accessing assets. Debugging and testing CPS is widely recognized as challenging, with various techniques being questioned for their effectiveness [9].

To provide a framework capable of real-time analysis, prompt mitigation, and minimal computational burden, this paper introduces a digital twin framework based on runtime verification. This framework seamlessly integrates data analytics with the learning of Linear Temporal Logic (LTL) formulas. The framework integrates machine learning algorithms to acquire LTL formulas from past data (training data). The paper's contribution lies in designing a process to generate system-specific LTL formulas using machine learning and implementing an LTL-based runtime verification cybersecurity framework for digital twin cyber-physical systems. This framework is applicable for tackling engineering/network data-related attacks where patterns can be identified in time series. The objective of this framework is to anticipate events that precede an adverse occurrence before it actually takes place.

2 The Proposed Approach

This section presents our Linear Temporal Logic (LTL) based runtime verification digital twin framework. The systematic approach is divided into five phases; Phase I (Data Pre-processing), Phase II (Data Clustering), Phase III (Domain Expert Analysis), Phase IV (LTL Formula Learning), and Phase V (Runtime Monitoring)

Phase I: Data Pre-processing Our framework begins with the collection of historical datasets and their pre-processing. In the process of learning Linear Temporal Logic (LTL) formulas from historical datasets, it is anticipated that the dataset encompasses both regular and anomalous events. The dataset is split into training and testing datasets. During this stage, the training dataset is used to train a model while to evaluate the model's performance after it has been trained, a testing dataset is used. With the aid of machine learning, we build a system model based on sample data, known as training data, to learn the LTL formula for the system.

First, the dataset is pre-processed which involves data cleaning, data transformation, feature selection or data reduction, handling missing data and data encoding. The data pre-processing depends on the data classification. To initiate the data pre-processing phase, we create a Python algorithm named *'LTL_Formula_Learner.py'* for learning Linear Temporal Logic (LTL) formulas.

Phase II: Data Clustering The next phase of our methodology is data clustering, an artificial intelligence process that 'learns', that is, leverages data for improvement of performance on some set of tasks. During this stage, the goal is to

recognize and categorize data into distinct clusters or groups. This process aims to distinguish clusters that correspond to favourable and unfavourable events, essential for the subsequent learning of patterns that will translate into Linear Temporal Logic (LTL) formulas.

In the case where the dataset is already labelled, the clustering algorithm may only be used to group column values into two variables 1 and 0 which is required for the generation of the LTL formula using the learning algorithm. For our research purpose, the K-means algorithm is used for data clustering.

Phase III: Domain Knowledge Expert Analysis The third phase of our methodology is domain expert analysis. Incorporating domain-specific knowledge and expertise to select features that are known to be significant for cyber security analysis is important and crucial. Subject matter experts can offer valuable insights into pivotal system components susceptible to data modifications or anomalies. These insights aid in selecting the most informative features. For our framework, a domain knowledge expert assists in identifying which of the clusters present normal and abnormal behaviour of the system.

Phase IV: LTL formula Learning The next phase of our approach is the LTL formula learning. At this phase, an LTL formula is generated based on the historical data set. In order to learn the LTL formulae, we implement the samples2LTL algorithm [6]. The objective of the algorithm is to acquire an LTL formula that distinguishes between two sets of traces: positive (P) and negative (N). The resultant formula should accurately represent every trace in the positive set (P) while not being applicable to any trace within the negative set (N).

The samples2LTL algorithm takes in an input file termed as traces separated as positives (P) and negatives (N) by $---$. Each trace is a sequence of states separated by ';' and each state represents the truth value of atomic propositions. An example of a trace is $1, 0, 1; 0, 0, 0; 0, 1, 1$ which consists of two states each of which defines the values of three propositions and by default considered to be $x0, x1, x2$.

For our framework, we learn patterns by analysing rows leading to bad events to predict events before happening, therefore, the samples2LTL algorithm takes in the trace file which contains the 'n' rows leading to bad events as a set of positives (P), and 'm' rows indicative of good events as a set of negatives (N). This is stored in the samples2LTL folder as 'example' with the extension '.trace'.

Phase V: Runtime Monitoring Runtime monitoring is the last phase of our approach. This is the process where the runtime checker verifies the real-time data against security properties for runtime checking. In runtime verification, the LTL formula is used to define a system property to verify the execution of the system.

In our previous work [4], we presented a runtime verification engine for the digital twin that can verify properties in multiple temporal logic languages. The runtime verification supports both FLTL and PTLTL in one package and is driven by the model checker Process Analysis Toolkit (PAT). In this paper,

we implement the runtime verification engine with the declaration of the LTL formula as the property. This paper adopts LTL on finite trace (FLTL) with strong next, that is, $\mathbb{X}\ A$ is true when the next state exists and makes A true; otherwise, $\mathbb{X}\ A$ is false. In this semantics, $\mathbb{F}\ A$ is only true when there is a future state that makes A true; otherwise, it is false. FLTL looks into the future. Also adopted in the paper is Past-time LTL (PTLTL), another useful language for specifying security-related properties [2] which has two distinct temporal operators called previously ($\mathbb{P}$) and since ($\mathbb{S}$). Their semantics are defined on past state traces, which are symmetric to FLTL. In PTLTL, $\mathbb{P}\ A$ is true when the previous state exists and makes A true; this is symmetric to $\mathbb{X}\ A$ in FLTL. $A\ \mathbb{S}\ B$ is true if 1) the current state makes B true, or if 2) B was true sometime in the past, and since then, A has been true. The semantics of $A\ \mathbb{S}\ B$ in PTLTL is symmetric to $A\ \mathbb{U}\ B$ in FLTL.

We incorporate our runtime validation through an algorithm called execute_runtime.py, which builds upon the foundation of our previous work's runtime-monitor script. The runtime monitoring process consists of three distinct phases; the digital twin modelling, property definition and the runtime verification.

Digital Twin Modelling: We model the system using the testing dataset initially set aside to evaluate the model's performance. This serves as our digital twin model which is modelled using PAT. In our approach, we are mainly interested in verifying properties over the state variables of the system. Let us name the state variables *var1, var2,* $\cdots$. A state S is simply a snapshot of the values of state variables, i.e.,$S ::= \{var1 = val1,\ var2 = val2,\ \cdots\}$. In PAT, we model a state via a process in Communicating Sequential Processes [3] with C# (CSP#) [7]. The process performs variable assignments as below.

$$S() = \{svar1 = val1; svar2 = val2; ...\} \rightarrow Skip;$$

A final trace T is a sequence of states, modelled as below.

$$T() = S1(); S2(); ...$$

Property Definition: The user can define properties over state variables. For example, the below code defines a proposition that states "var1 is not 0."

$$\#define \quad v1Safe \quad (var1 != 0);$$

We can then use PAT to check a safety property that "var1 should never be 0" using the temporal modality G, which is written as [].

$$\#assert \quad Trace() \models []v1Safe;$$

Verification: Given the generated LTL formula from the historical data, the property is defined for the runtime verification as the safety property in Process Analysis Toolkit (PAT) language. The foundation of our runtime verification

framework is based on the observation that verifying LTL with finite traces in PAT language corresponds to verifying FLTL with strong next/future.

Utilizing this framework, we can identify data-related attacks that exhibit transient patterns in time series. This approach can be deployed in various domains, including cyber-physical systems (CPS), the metaverse, satellite communication systems, and the Internet of Things (IoT). Due to limited space, this paper focuses on a single case study related to a water distribution system.

3 Case Study: Water Distribution System

This is a main water distribution system operator of C-Town and the dataset was created and published by the BATADAL team [8]. C-Town consists of 388 nodes linked with 429 pipes and is divided into 5 district-metered areas (DMAs). The SCADA data include the water level at all 7 tanks of the network (T1–T7), the status and flow of all 11 pumps (PU1–PU11) and the one actuated valve (V2) of the network, and pressure at 24 pipes of the network that correspond to the inlet and outlet pressure of the pumps and the actuated valve. Three distinct datasets from the system generated. However, for our specific application, we focused our analysis on "training_dataset_2.csv". This dataset, which includes partially labeled data, was made available on November 28, 2016. It spans approximately six months and encompasses several attacks, some of which have approximate labels.

In the dataset are 43 columns, attack labelled using a column named 'ATT_FLAG' with a 1/0 label column, with 1 meaning that the system is under attack and 0 meaning that the system is in normal operation. After collating the dataset [8], we implement our framework to learn a pattern from the dataset indicative of the attack carried on the system. Using the observed pattern, we learn the LTL formula for the system.

The LTL runtime verification-based digital twin framework algorithm developed for this work can be accessed at the following link: `https://github.com/deejay2206/LTL-based-Runtime-Verification`. For further references on the LTL formula learning algorithm used in our work, see samples2LTL.

LTL-Formula Learning: Using the samples2LTL algorithm, we generate a list of the LTL formula as shown below. The LTL formula is inputted as the LTL property which is used in the runtime checker. We define the LTL formula in PAT as property.csp.

```
(x20 U x37), X(x39), !(x10), !(x5), F(x28);
```

Conducting runtime verification involved generating a digital twin model of the system using the testing dataset. This dataset was divided into distinct traces, each of which represented a model. These models were then fed into the runtime checker, as described in Section 2, to validate the adherence of the learned LTL formula set as a system property.

Result Analysis To access the performance of our framework, we use evaluation metrics and to achieve this, we use the confusion matrix which is the calculation of number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) based on the predictions and the actual outcomes. In our data analysis, we considered each row within the dataset as an individual event. There were a total of 263 positive traces and 989 negative traces, resulting in a combined count of 1252 events or instances. We calculate the accuracy and precision. The result revealed TP = 242, TN = 901, FP = 54, FN = 55. The outcome demonstrated a 91% percent accuracy rate and the positive predictive value is 82% for the predictive capabilities of our framework.

4 Conclusion

Engineering or network data related attack leading to the temporal pattern of behaviour of a real critical infrastructure can cause great harm to a human being. With the concept of a runtime-based digital twin system, this temporal pattern of behaviour can be detected. In this paper, we investigated how to learn the LTL formula from historical data and evaluated our approach using a case study in cyber-physical systems.

References

1. Baheti, R., Gill, H.: Cyber-physical systems. The impact of control technology **12**(1), 161–166 (2011)
2. Du, X., Tiu, A., Cheng, K., Liu, Y.: Trace-length independent runtime monitoring of quantitative policies. IEEE Transactions on Dependable and Secure Computing **18**(3), 1489–1510 (2019)
3. Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM **21**(8), 666–677 (1978)
4. Hou, Z., Li, Q., Foo, E., Song, J., Souza, P.: A digital twin runtime verification framework for protecting satellites systems from cyber attacks. In: 2022 26th International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 117–122. IEEE (2022)
5. Hu, J., Lennox, B., Arvin, F.: Robust formation control for networked robotic systems using negative imaginary dynamics. Automatica **140**, 110235 (2022)
6. Neider, D., Gavran, I.: Learning linear temporal properties. In: 2018 Formal Methods in Computer-Aided Design (FMCAD). pp. 1–10. IEEE (2018)
7. Sun, J., Liu, Y., Dong, J.S., Pang, J.: Pat: Towards flexible verification under fairness. In: International conference on computer aided verification. pp. 709–714. Springer (2009)
8. Taormina, R., Galelli, S., Tippenhauer, N.O., Salomons, E., Ostfeld, A., Eliades, D.G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M.K., et al.: Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. Journal of Water Resources Planning and Management **144**(8), 04018048 (2018)
9. Zheng, X., Julien, C., Kim, M., Khurshid, S.: Perceptions on the state of the art in verification and validation in cyber-physical systems. IEEE Systems Journal **11**(4), 2614–2627 (2015)