An automata theoretic framework for analyzing effects of code replacement attacks on scheduling

***Abstract*—Cyber Physical Systems (CPS) such as a manufacturing plant, a power generator, a power transmission substation etc. are usually controlled by a Supervisory Control and Data Acquisition System. Due to the rise of global terrorism, and cyber criminals, these systems are ripe targets of cyber attacks. Among the various attack surfaces recognized by the community, the possibility of replacement of previously vetted control software, or other software components in the system by malicious variants by insider attackers is an acute possibility. In this paper, we propose an automated framework that addresses the effect of code replacement attacks from the perspective of loss of control performance. Given a set of control components, a control objective to be satisfied by the control ensemble, the question of schedulability and synthesis of a scheduler that can ensure the desired control performance has been recently studied in literature. In this paper, we extend the same philosophy to build an automata theoretic framework for assessment of code replacement attacks on schedulability. We have built an end-to-end framework that takes in a set of control components, their variants (after code replacement), a control objective (e.g. Rabin's Acceptance condition) to be guaranteed, and performs an automated schedulability assessment. We report some preliminary experiments of our framework on simple benchmarks.**

***Keywords*-CyberPhysical System, Cooperative Control, Code Replacement Attack, Insider threat, Rabin's Acceptance Condition, ω-regular language, Büchi Automaton, Schedulability**

I. Introduction

Most critical infrastructures such as power grid, water / sewage control, power generation plants, industrial automation etc., are cyber-physical systems (CPS). Cyber-Physical systems have two major interacting components – a component with physical dynamics governed by laws of physics – modeled with partial differential equations, another component – control, computing and networking which controls the physical dynamics. For very large and geographically distributed cyber physical systems, the control trajectory is designed to be stable and reliable with very complex distributed as well as centralized control. The control components today routinely involve a non-trivial amount of software, running on embedded control units on-board, to support and run any moderately sophisticated CPS infrastructure.

In recent history, cyber physical systems have been a popular victim of choice for so called cyber attacks, the effects of which have been moderate to as ravaging as blackouts [1] or financial loss. This has prompted NIST [2] to promote cyber security frameworks for critical infrastructure [3] as one of the themes of immediate research attention. Given that no cyber security framework is full proof, significant research thrust is being invested in recent times to develop techniques that allow us to continually monitor the physical dynamics of these systems and look for any anomalies that could be indicative of cyber attack induced problems. Early detection of system dynamics changes would allow us to contain the damage by immediately islanding parts of the system which seem to be the origin areas in the infrastructure.

Research investment in cyber security of CPS is extremely crucial for developing novel technology for cyber attacks detection, prevention, and countermeasures. Systematic studies on different sources of cyber-attacks have revealed myriads of possibilities by which these cyber threats can propagate inside a CPS infrastructure. A popular and widely acknowledged means of cyber-attack, *phishing attacks*, originate when someone whose desktop is connected to the control network opens an email containing a payload, which can then take over the control of the business network, and in turn the control network. However, since much of the cyber threat models also assume insider knowledge or sabotage, even an isolated control network is not devoid of cyber-attack possibilities. A popular example is the notorious Sony Pictures hack [4] which leaked a release of confidential data from the film studio Sony Pictures. A person with local or remote access to the equipment of the physical plant, or access to the various interfaces such as programmable logic controllers (PLCs) or other Intelligent Electronic Device (IEDs) that are connected to the physical system for measurements and control can exploit a vulnerability in these devices to induce an attack on the system. One could also gain access to the control network, and create various kinds of man-in-the-middle attacks by either suppressing measurements or control actuation signals, replaying stale measurements or actuation signals, or even injecting maliciously planned false data. These kinds of attacks would then mislead the controllers, and wrong control actions could lead to disastrous industrial accidents. One could also hack into the controllers or the various other computing elements in the control center such as the process control servers by exploiting some vulnerabilities in their design and attack the cyber physical system. In fact, in case of the Stuxnet worm [5], vulnerabilities in the Siemens SCADA system were exploited. While individual areas of cyber security research have received much attention in academia (e.g. cryptography; crypto-analysis; network security in the form of firewall and other perimeter security techniques, and intrusion detection, anti-virus software, and static analysis of software to detect vulnerabilities and zero-day attacks; hardware Trojan detection) much of these research focus on guarding IT systems in business and corporate information infrastructure.

The focus of study in this paper is code replacement attacks, wherein some or parts of a control component are compromised by tampering their operational modality by modifying the software running inside. Code replacement [6] by internal employees, phishing attacks or other means of code injection have been known to be a vector for cyber-attacks for a while. Stuxnet analysis showed that such attacks were part of the repertoire in that case. In an industrial control environment with real time control components, code replacement can lead to disaster, for example, slowing down a particular process in the industrial manufacturing can cascade a chain of failures in the whole assembly line. In this paper, we propose an idea that will demonstrate that such attacks for real-time SCADA systems can be guarded against by statically analyzing the legitimate control programs, and constructing an omega-regular language based timing signature, which can then be periodically checked on the running components to distinguish a replaced component from the original component. The timing signature analysis of omega-regular languages was originally proposed by researchers in [7], in the context of real-time communication scheduling in SCADA systems. In this paper, we plan to take the idea further for addressing the problem of guarding against code replacement attacks on real-time control systems. Given a set of control components, a control objective to be satisfied by the control ensemble, the question of schedulability and synthesis of a scheduler that can ensure the desired control performance has been recently studied in literature [7], [8], [9]. In this paper, we extend the same philosophy to build an automata theoretic framework for assessment of code replacement attacks on schedulability.

This paper is organized as follows. Section III describes the background theory for this work, while Section IV formally defines our problem statement. Section V presents the solution architecture. In the following discussion, Sections VI and VII present some illustrative examples and experiments respectively. Section VIII concluded this discussion.

II. Related work

Weiss et al[10], depicts the idea of automata based control scheduling. They expressed the communication interface among the control components by using formal language. In their paper they have illustrated how the interfaces of discrete-time switched linear system can be expressed using finite Büchi automaton. [8] have described CPU scheduling in terms of finite states over infinite words. This infinite words depicts the particular time slot for which a particular resource can be allotted to CPU. Weiss et al[7] has also applied automata based scheduling on network, to formalize the effect of bus scheduling and stability of the network. Further, [9] describes how the interfacing among the control actions can be modeled by Büchi automaton to guarantee stable and reliable scheduling. In this paper they have demonstrated, how to construct the scheduler automata where each state represents one particular control schedule and generated the permissible schedule in terms of ω-regular language.

III. Background

In this section, we present two background concepts that are necessary to establish the foundation of this work. We begin with the Büchi automaton.

Büchi Automaton: A Büchi automaton [11] is described as a five tuple, $A = (Q, I, \delta, q_0, F)$, where Q is a finite set of states, I is the input alphabet, $\delta : Q$ x $I \rightarrow Q$ is the transition function, q_0 is an initial state ($q_0 \in Q$) and F is a set of final states. An infinite word $\delta \in I^{\omega}$ over the input alphabet takes the automaton through an infinite sequence of states $q_0, q_1, q_2, ...$, which describes a run of the automaton such that $q_{k+1} \in \delta(q_k, \delta_k)$ where $k \in \mathbb{N}$ and δ_k refers to the k^{th} symbol on the input word δ. An infinite word is accepted if some accepting state $q_f \in F$ appears in the run $q_0, q_1, q_2, ...$ infinitely often. If $|\delta(q, i)| = 1$ where $q \in Q$ and $i \in I$, then it is a deterministic Büchi automaton, otherwise it is non-deterministic.

IV. Problem statement and Motivating Example

The motive of our work is to detect code replacement attack. Code replacement attack signifies some changes to the system. We are trying to capture the results of the attack from the perspective of scheduling. Here in this work, we are examining how this changes to the system effects the schedulability of the system.

To explain our problem we are considering a system consisting of many control applications. Each of these control applications is partitioned into a set of tasks. Each of these tasks are mapped onto different processors. These tasks of the controllers communicates via shared buses. The tasks on each processors are scheduling using a given scheduling policy. The messages on the shared buses are also scheduled using a given arbitration policy. When a process is executed by processor it goes through a number of states before termination. Among all these states some states require bus access. Those states are allocated to the bus when their term comes.

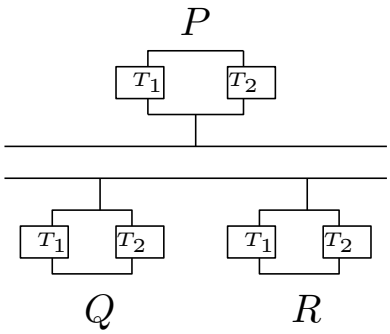

Fig. 1. *Control Loops sharing bus*

The diagram depicted in Fig.1 comprising of three control loops P, Q and R. Each control loop has two tasks T_1 and T_2. Here we are considering, from each of these tasks at least one state requires bus access. Let us consider p_0 and p_1 from

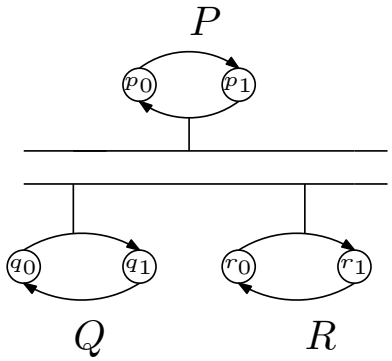

Fig. 2. *Automata reprenting Bus accessing states*

control application P, q_0 and q_1 from control application Q and r_0 and r_1 from control application R are the bus access states. The state diagram representation of these control loops is depicted in Fig. 2.

As we have stated earlier that we are trying to observe the effect of code replacement attack in terms of scheduling sequence, our system will be examinig the scheduling sequence after a particular interval of time. Say, our expectation is that at every point of time we want to obtain a scheduling sequence where at least one bus accessing state from each control application is present. In this work such scheduling expression are represented by ω-regular language. Extraction of such an expression for the automaton given in Fig. 2 is depicted in Fig. 3.

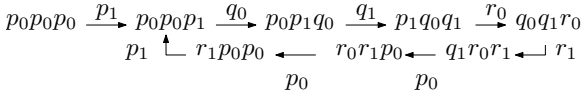

Fig. 3. *Cycle construction and ω-regular expression extraction*

From the cycle depicted in Fig.3 we can extract the ω-regular expression $(p_1q_0q_1r_0r_1p_0p_0p_1)^\omega$.

Let us consider, if a code replacement attack take place on this system and one of the control applications has lost the bus access. As a result of this attack no state from this finite state machine will take part in bus access that we can capture by checking the scheduling sequence. The scheduling sequence will not contain any state of this replaced control application as a consequence of the attack.

Let us consider, control application P got effected due to code replacement attack, bus accessing states are not getting the bus access. The ω-regular expression will be depicted in Fig.5 :

From the cycle depicted in Fig.5 we can extract the ω-regular expression $(r_0r_1q_1q_0q_0q_0q_0)^\omega$. The expressin does not contain any state from the control application P.

On the other hand, if it would happen that after code replacement attack instead of the earlier state some other state is accessing the bus and the earlier ones (those who were accessing the bus before the attack took place) lost their bus access then our expectation cannot capture that the attack has taken place.

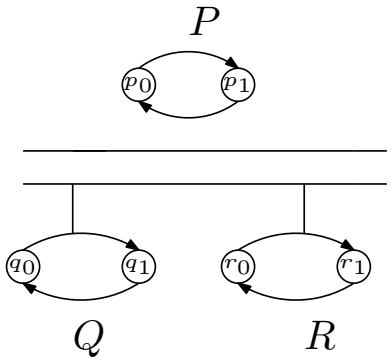

Fig. 4. *Automata reprenting Bus accessing states after code replacement attack*

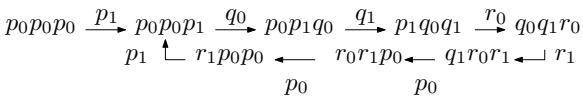

Fig. 5. *Cycle construction and ω-regular expression extraction*

V. Solution Architecture

Approaches to solve the above stated Problem:

There are n number of control actions. At least one states from each of them are sharing a common bus. Let , k number of states are sharing a common bus. In this paper, we are proposing an idea to detect code replacement attack. After a particular interval of time the system will check the schedulability of the system to assure that code replacement attack has not taken place. If at least one bus accessing state from each control application is not observed in ω-regular exression, then we can say that code replacement attack has taken place. The following steps represent the necessary steps to solve the problem:

1) Formation of the control sequence: Generate all k^L possible control schedule sequences up to a certain length.
2) A tree is formed with scheduling states where each path describes one schedule. The leaf nodes represent the state comprising of the path sequence.
3) Construct a Büchi automaton with these resulting states: Considering the set of states of a Finite State Machine, an automaton is constructed. Here each state is accepting state. After constructing the transition matrix if a cycle can be obtained over the states then it is a Büchi Automaton.
4) After the construction of the Büchi automaton ω-regular expression is extracted. If the expression does not contain at least one bus accessing state from each control application then we can say code replacement attack has taken place

VI. Illustrative examples

In this section we illustrate our approach considering the example given in Fig.2. Three control applications are sharing the bus , only the bus accessing states are represented in the diagram. According to the steps given in the solution architecture section we constructed the automaton. Here we have 6 states competing for the bus access from three control applications. If we choose the length of the control schedule length to be 3 then we have $6^3 = 216$ number of sequneces. The way the tree can be formed is depicted in the Fig.6

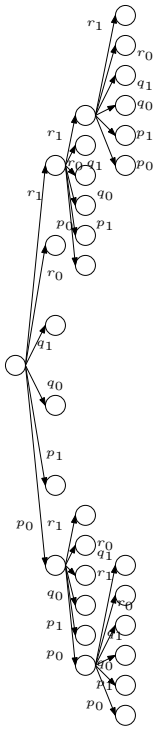

Fig. 6. *Control Loops sharing bus*

The leaf nodes represents the state for the Büchi automaton. Each path of the tree represents one possible scheduling sequence. The leaf node represents the sequence when they take part in Büchi automaton construction. The next step is the construction of Büchi automaton. With all leaf nodes, all possible transitions are added and the newly formed state is matched with all other states including itself. If any matches found, those transitions are added for the finite state machine.

After construction of the automata the next step is to find out at least one cycle from the automaton which contains at least one state from each control application. If the expected result is not met we can say the code replacement attack can happen.

VII. Experiment

VIII. Conclusion and Future Work

There has been a lot of work on automaton based control scheduling but we have used this idea to detect code replacement attack. In this paper we have demonstrated how schedulability of control actions can be a key issue to detect code replacement attack. We have proposed that schedulable system can no longer remain schedulable if code replacement attack take place. To detect code replacement attack, the schedulability of the system will be checked repeatedly to check if any control actions has been hijacked or not.

Our aim is to prepare a tool in future so that it can be scalable and be checked against many control actions.

References

[1] "Report on the grid disturbances on 30th july and 31st july 2012," National Institute of Standards and Technology, Tech. Rep., August 2012.

[2] "National institute of standards and technology." [Online]. Available: http://www.nist.gov

[3] "Framework for improving critical infrastructure cybersecurity," National Institute of Standards and Technology, Tech. Rep., February 2014.

[4] "Sony pictures hack." [Online]. Available: https://en.wikipedia.org/wiki/Sony_Pictures_hack

[5] "The stuxnet worm." [Online]. Available: spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet

[6] S. Ghosh, J. Hiser, and J. W. Davidson, "Replacement attacks against vm-protected applications," in *Proc. of VEE*, 2012, pp. 203–214.

[7] G. Weiss, S. Fischmeister, M. Anand, and R. Alur, "Specification and analysis of network resource requirements of control systems," in *Proc. of HSSC*, 2009, pp. 381–395.

[8] R. Alur and G. Weiss, "Regular specifications of resource requirements for embedded control software," in *Proc. of RTAS*, 2008, pp. 159–168.

[9] S. K. Ghosh, A. Mondal, S. Dutta, A. Hazra, and S. Dey, "Synthesis of scheduler automata guaranteeing stability and reliability of embedded control systems," in *Proc. of VDAT*, 2016, pp. 127–132.

[10] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Proc. of HSCC*, 2007, pp. 601–613.

[11] W. Thomas, "Automata on infinite objects," in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 1990, pp. 133–192.

[12] M. S. Prabhu, A. Hazra, and P. Dasgupta, "Reliability guarantees in automata-based scheduling for embedded control software," *Embedded Systems Letters*, vol. 5, no. 2, pp. 17–20, 2013.

[13] M. S. Prabhu, A. Hazra, P. Dasgupta, and P. P. Chakrabarti, "Handling fault detection latencies in automata-based scheduling for embedded control software," in *Proc. of CACSD*, 2013, pp. 1–6.

[14] M. Zamani, S. Dey, S. Mohamed, P. Dasgupta, and M. M. Jr., "Scheduling of controllers' update-rates for residual bandwidth utilization," in *Proc. of FORMATS*, 2016, pp. 85–101.