AFRL-AFOSR-JP-TR-2018-0035

CORESAFE:A Formal Approach against Code Replacement Attacks on Cyber Physical Systems

Sandeep Shukla
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

04/19/2018
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOA
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
20-04-2018	Final	20 Sep 2016 to 19 Sep 2017

4. TITLE AND SUBTITLE
CORESAFE:A Formal Approach against Code Replacement Attacks on Cyber Physical Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA2386-16-1-4099

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Sandeep Shukla

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
IIT Campus KANPUR POST
Kanpur, 208016 IN

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-JP-TR-2018-0035

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Industrial Control Systems (ICS) used in manufacturing, power generators and other critical infrastructure monitoring and control are ripe targets for cyber-attacks these days. Examples of such attacks are abundant such as attacks on Iranian nuclear enrichment plant with Stuxnet in 2009, on German steel plant in 2014, Ukrainian power system in 2015 and 2016. Usually in ICS, multiple control loops work concurrently and share various resources including the communication bus through which they interact with sensors and actuators. Real-time scheduling of concurrent control applications while competing for shared resources demands a delicate balance between performance and real-time constraints. A possible insider attack could be the replacement of a previously vetted control application or other components in the system, during a system update. In this project, we worked on an automated framework that addresses the effect of such 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 project, we extended this idea further to build an automata theoretic framework for assessment of replacement attacks on schedulability. We have built an end-to-end framework that takes in a set of control components, their variants (after replacement), a control objective to be guaranteed, and performs an automated schedulability assessment. We report some preliminary experiments of our framework on simple benchmarks.

15. SUBJECT TERMS
Code Replacement Attacks, Cyber Physical Systems, Formal Approach

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			SERNA, MARIO
Unclassified	Unclassified	Unclassified	SAR	15	19b. TELEPHONE NUMBER *(Include area code)* 315-227-7002

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Final Report for AOARD Grant "CORESAFE: A Formal Approach against Code Replacement Attacks on Cyber Physical Systems"

Sandeep K. Shukla, Debleena Das, Anshuman Banerjee
Indian Institute of Technology Kanpur
India

Contents

Summary: 2
Introduction: 2
  Problem addressed in this project: 4
  Contributions of this work: 4
Related Work: 5
Experiment 5
  Problem Model: 5
  Schedulability violation attacks: 6
  Solution Architecture: 6
  Intersection Automata construction: 8
  Verifying the scheduling objective: 8
Results and Discussion 9
Conclusion and Future Work 10
Publication from this project: 12

Final Report for AOARD Grant "CORESAFE: A Formal Approach against Code Replacement Attacks on Cyber Physical Systems"

Sandeep K. Shukla, Debleena Das, Anshuman Banerjee
Indian Institute of Technology Kanpur
India

PI and Co-PI information: SANDEEP KUMAR SHUKLA, Indian Institute of Technology Kanpur, India

Period of Performance: September 20, 2016 – September 19, 2017

Summary:

Industrial Control Systems (ICS) used in manufacturing, power generators and other critical infrastructure monitoring and control are ripe targets for cyber-attacks these days. Examples of such attacks are abundant such as attacks on Iranian nuclear enrichment plant with Stuxnet in 2009, on German steel plant in 2014, Ukrainian power system in 2015 and 2016. Usually in ICS, multiple control loops work concurrently and share various resources including the communication bus through which they interact with sensors and actuators. Real-time scheduling of concurrent control applications while competing for shared resources demands a delicate balance between performance and real-time constraints. A possible insider attack could be the replacement of a previously vetted control application or other components in the system, during a system update. In this project, we worked on an automated framework that addresses the effect of such 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 project, we extended this idea further to build an automata theoretic framework for assessment of replacement attacks on schedulability. We have built an end-to-end framework that takes in a set of control components, their variants (after replacement), a control objective to be guaranteed, and performs an automated schedulability assessment. We report some preliminary experiments of our framework on simple benchmarks.

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 and schedules them for execution. 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

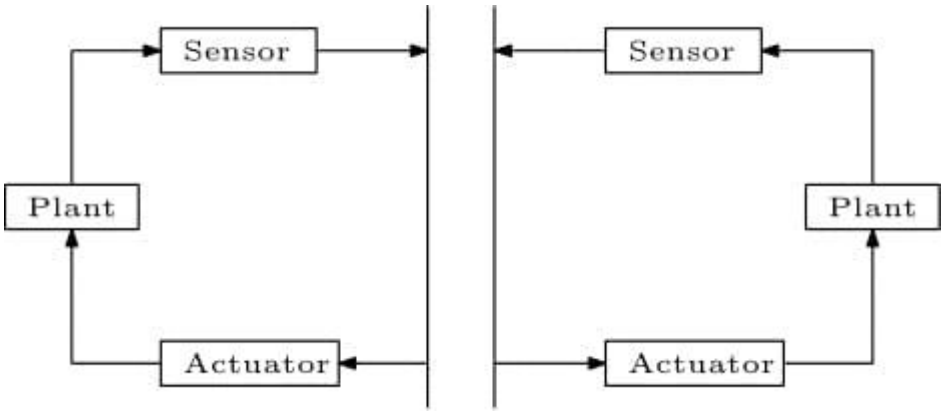

Fig. 1: Control loops

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 point to possible origin areas in the infrastructure.

Research in security of CPS is extremely crucial for developing novel technologies for cyber-attack 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 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 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 vulnerabilities in their design and attack the cyber physical system. In fact, in case of the Stuxnet worm [4], 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.

Problem addressed in this project:

In this work we have studied the CPS security problem from a different perspective: the focus of study in this work is schedulability attacks, wherein some or parts of a control component are compromised, and the scheduler is unable to schedule the different components in any way to meet the control objectives of the CPS. In an industrial control environment with real time control components, this can lead to disaster since some of the components may malfunction due to lack of input from the scheduler. As an example, slowing down a particular process in industrial manufacturing can cascade a chain of failures in the whole assembly line. In this work, we proposed 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 check for schedulability. The idea presented here is based on the timing signature analysis for omega-regular languages [5], in the context of real-time communication scheduling in CPS.

Contributions of this work:

During the course of this project, we addressed the problem of detecting component replacement attacks from the schedulability perspective. 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 [5],[6], [7]. In this work, we extended the same philosophy to build an automata theoretic frame-work for assessment of replacement attacks on schedulability. The foundation of our attack analysis framework is based on the notion of infinite automata-based reasoning of control performance and schedulability analysis, as illustrated in the following section. Automata theoretic modeling frameworks have been quite popular in literature for a wide variety of applications. Additionally, the power of finite automata over infinite words (Büchi automata in particular) has been exploited in recent literature in control performance and stability analysis. Our work is another step in the same direction for assessing the effect of replacement attacks in cyber-physical control.

Related Work:

Replacement attacks [8] carried out by internal employees, phishing attacks, code injection attacks 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. The work in [9] describe the idea of replacement attack on malware behavior. Malware belonging to same family are supposed to show similar behavior but if the replacement attack changes the behavior specification, then in spite of being from same family, their behavior will be different. Malware interact with operating system through system call. A system call dependency graph(SCDG) shows the data flow dependency among system call. In this paper, they propose the idea of SCDG replacement and the after replacement effect on the malware behavior. To detect the attack, malware analysts are required to put extra efforts to re-analyze the similar samples. Further, [10] represents how the protected PVMs(Process-level virtual machines) can be replaced with attack VM(virtual machines) which makes the application more vulnerable to analysis and modification. The analysis of replacement attack on generic watermarking system described in [11].

The idea of automata based control scheduling has been discussed in [12]. They express the communication interface among the control components using a formal language. This work illustrates how the interfaces of discrete-time switched linear systems can be expressed using a Büchi automaton. Authors in [6] describe CPU scheduling in terms of finite machines over infinite words. The infinite words depict the particular time slot for which a particular resource can be allotted to a specific control component. The work in [5] applies automata based scheduling on networks, to formalize the effect of bus scheduling and network stability. Further, [7] describes how the interfacing among the control actions can be modeled by Büchi automata to guarantee stable and reliable scheduling. This work presents a methodology for construction of a scheduler automaton where each state represents one particular control schedule while generating the permissible schedule in terms of a ω-regular language. Our work is an application framework based on the contributions above, with specific focus to component replacement attacks and schedulability analysis. While one of the major control objectives discussed in the above contributions has been exponential stability, which we assume to be satisfied for individual control components, we propose to use a fairness schedule in our work, that requires a different modeling strategy, as explained in the following. More importantly, our work here can be considered as an end-to-end framework on automata-based control performance modeling and assessment.

Experiment

Problem Model:

To explain our problem, we consider a control application system consisting of an ensemble of concurrently active control applications / loops. Each control application is partitioned into a set of tasks. The control tasks communicate via a shared bus. The messages on the shared bus are also scheduled using a given arbitration policy. When a control loop executes, it carries out the set of tasks as specified by different states in its control state machine. Among all these states, some states require bus access, for which the control loop presents a request to the bus arbiter. These

requests are considered by the bus arbiter / scheduler which decides on a bus scheduling strategy to grant bus access to the different control loops over time at different time slots.

As discussed in recent literature, we consider each participating control application is modeled as a Büchi automaton [13]. Büchi automata have been a popular mechanism of choice for modeling applications that require finite automatons over infinite strings or words. The basic structure of such automata is similar to their finite counterparts, with the exception of the acceptance conditions, as described below.

Definition 1. *A Büchi automaton is described as a five tuple, $A = (Q, \mathbf{I}, \delta, q_0, F)$, where Q is a finite set of states, $\mathbf{I}$ is the input alphabet, $\delta: Q \times \mathbf{I} \rightarrow Q$ is the transition function, q_0 is an initial state ($q_0 \in Q$) and F is a set of final states that model an infinitary acceptance condition.*

Definition 2. *An infinite word $\lambda \in \mathbf{I}^{\omega}$ over an input alphabet $\mathbf{I}$ takes a Büchi automaton $A = (Q, \mathbf{I}, \delta, q_0, F)$ through an infinite sequence of states $q_0, q_1, q_2, \ldots$, which describes a run of the automaton such that $q_{k+1} \in (q_k, \lambda_k)$ where $k \in N$ and λ_k refers to the k^{th} symbol on the input word λ. An infinite word is accepted by A if some accepting state $q_f \in F$ appears in the run $q_0, q_1, q_2, \ldots$ infinitely often.*

Each control loop is designed with a control objective in view, and the overall functioning of the system is dependent on the individual control loops meeting control objectives, along with a strategy for co-operative control of the shared message bus through which the control loops communicate with each other. We consider one such control system functioning correctly if both the above conditions are met. Meeting control objectives has been addressed in several recent articles, and is therefore, not discussed here. In contrast, we focus more on the schedulability aspect, as described below.

Schedulability violation attacks:

We now characterize the schedulability requirement below.

Definition 3. A set of concurrent control loops, each expressed as a Büchi automaton, is schedulable if there exists a strategy to schedule the individual control loops on the shared bus infinitely often. The scheduling objective is defined in the lines of infinitary acceptance, as is the case with Büchi objectives.

Intuitively, a scheduler should be able to allocate bus access to each individual control loop infinitely often. More specifically, given any infinite run of the composed system, at least one constituent bus accessing / accepting state from each control loop should repeat infinitely often. We consider a system safe if this holds, unsafe otherwise. We conclude that a replacement attack has been carried out if a system, initially safe and schedulable, turns out to be non-schedulable.

Solution Architecture:

In this section, we explain our solution architecture for schedulability assessment. As described in the previous section, we are given a set of *n* control loops, with their bus access states marked as the accepting ones. Recall that our scheduling objective is to be able to grant bus access infinitely often to each control loop. To check if the system is safe and schedulable, we carry out the following steps:

- We compute the product of the individual control loops using the construction explained below.
- Once the product is computed, we look for the presence of a cycle that contains at least one state from each control loop. In other words, on any infinite run of the product automaton, each control loop repeats infinitely often and is therefore, granted bus access infinitely often as well.

Definition 4: Intersection of Büchi automata[21]:

- We can build an automaton for $L(B_1) \cap L(B_2)$ as follows
- $B_1 \cap B_2 = (\Sigma, Q_1 \times Q_2 \times \{0,1,2\}, \Delta, Q_1^0 \times Q_2^0 \times \{0\}, Q_1 \times Q_2 \times \{2\})$
- We have $(\langle r, q, x\rangle, a, \langle r', q', y\rangle) \in \Delta$ iff the following conditions hold:
  - $(r, a, r') \in \Delta 1$ and $(q, a, q') \in \Delta 2$
  - The third component depends on the accepting conditions of B_1 and B_2.
    - If $x = 0$ and $r' \in F_1$ then $y = 1$.
    - If $x = 1$ and $q' \in F_2$ then $y = 2$.
    - If $x = 2$ then $y = 0$.
    - Otherwise, $y = x$
- The third component essentially guarantees that accepting states from both B_1 and B_2 appear infinitely often.

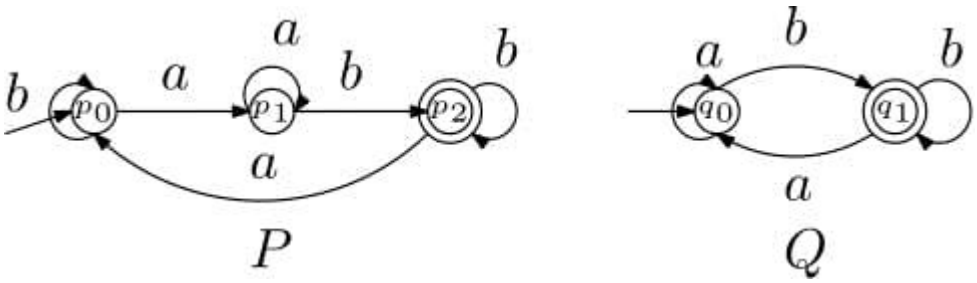

Fig. 2: Individual automaton

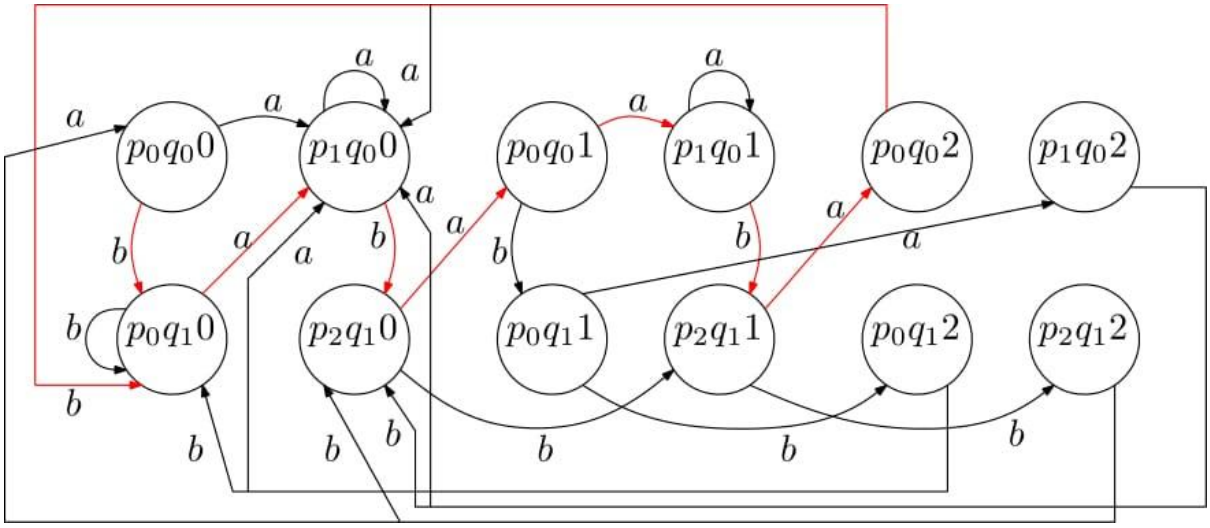

Fig. 3: Intersection of the input automata

Intersection Automata construction:

For the sake of simplicity and ease of illustration, we explain the intersection construction in terms of two automata. Let us consider the two Büchi automata *P* and *Q*, as shown in Fig.2 and defined by the conventional five tuple:

$P = \{(p_0,p_1,p_2), (a, b), p_0, Q \times \Sigma \rightarrow Q, p_2\}$

$Q = \{(q_0,q_1), (a, b), q_0, Q \times \Sigma \rightarrow Q, q_1\}$

From the diagrams given in Fig.2, we see that the automaton *P* can accept $(ab)^{\omega}$ and the *Q* automaton can accept $(b)^{\omega}$. The resulting intersection automaton will accept $(ab)^{\omega}$.

In the resulting automaton, a cycle is accepted, when it passes through the last copy of the product automaton that signifies the run of the cycle traverses the accepting states of each automaton infinitely often. The remaining cycles, which do not pass through the last copy, are not accepted. We thus augment the classical product construction for infinite automata to model our infinitary acceptance requirement.

Verifying the scheduling objective:

Once the intersection automaton is constructed, the task of looking for cycles starting from the initial states is carried out. Cycle detection algorithms are standard in literature [15] and we implement the same for this work as well. Once a cycle is found, we traverse the cycle (every cycle is bound to contain a finite number of states) and check if each control loop is represented inside it. If not, we proceed to the next cycle and carry out the same check. If all cycles are exhausted and we do not encounter any one which meets our scheduling objective, we conclude that the schedulability requirement is not met. This step is thus quite straightforward.

Consider the example given in Figure 4. It can be seen that there are multiple cycles in the intersection automata, some of which do not contain one representative accepting state from each control loop (e.g. <p0, q0, 0 > → < p1, q0, 0 > → < p1, q1, 0 >). Thus it is necessary to examine each cycle to check for existence of at least one accepting state from each control loop, which is the main idea behind the notion of schedulability we adopt here.

F. Schedulability violation attack detection:
As the system is monitored over time, newer structures of the control loops emerge and we perform the same computation steps outlined above on the modified structures to check if it remains schedulable even in the presence of the modifications. Our mechanism can thus be used as a continuous monitor to safe-guard against intermittent control attacks.

At this onset, we would like to admit that our framework can point out replacement attacks only if schedulability is lost. Once the intersection is computed, we may have 3 different possibilities if schedulability is lost. Assuming that the system was schedulable earlier, we may conclude the system has been attacked if any of the following possibilities are detected. Firstly, we may have an empty intersection automaton now. As a second possibility, the product may still be non-empty but a cycle may not be reachable. Finally, a cycle may be found but it may not contain representative tasks from each control loop. All these are indicators of schedulability loss according to our analysis method and we conclude that a replacement attack is carried out. However if the system still remains schedulable according to our scheduling objective, our framework is unable to flag the attack, even if a replacement has been carried out.

Results and Discussion

We have built an end-to-end tool in Python for code replacement attack detection. The tool takes in a set of control loop descriptions, computes their intersection and implements the cycle detection step. The tool has been applied to a number of small examples of synthetic control applications and their variants. Due to the lack of standard open source benchmarks in this domain, we have performed all our experiments on synthetic benchmarks of various representative sizes, varying the number of individual control loops, and the number of constituent states of each. For each row, for a fixed number of automata, we took up the initial descriptions and created random attacks by modifying some of the components, and carried out our analysis. As expected, our tool was able to flag out the schedulability violation in each of the cases.

The following table represents the overall details of our experiments. Column 2 mentions the number of components participating in the control application, while the next 2 columns show the result of the schedulability check and the time taken to perform that. Columns 5 and 6 show the performance of the tool for detecting the attack. With increase in the number of participating components, the time for checking schedulability also increases, as can be seen from the final column.

Serial No.	No. of automata	Safety check (before attack)	Scheduling time in seconds	Safety check(after attack)	Scheduling time in seconds
1	2	YES	0.00069	NO	0.00038
2	4	YES	0.00354	NO	0.00054
3	6	YES	0.00661	NO	0.00920
4	8	YES	0.02248	NO	0.00693
5	10	YES	0.23043	NO	0.14550
6	12	YES	0.69302	NO	0.67120
7	14	YES	1.78644	NO	1.74657
8	16	YES	4.08354	NO	3.90508

Conclusion and Future Work

In this work, we present an early foundation of a framework for schedulability violation attack detection for cyber-physical systems. We adopt standard techniques to build an automata-based control analysis foundation. We have built an end-to-end tool that implements the proposed ideas. We are currently examining the applications of other infinitary acceptance conditions (e.g. Rabin, Muller) in this domain, with respect to their ability to detect different attack scenarios. Additionally, we are also exploring new algorithms for intersection automaton construction that might prove helpful as the number of control loops grow. We believe that our initial results will open up more interesting avenues in this research.

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

[5] 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.

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

[7] 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.

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

[9] J. Ming, Z. Xin, P. Lan, D. Wu, P. Liu, and B. Mao, “Replacement attacks: Automatically impeding behavior-based malware specifications,” in Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers, 2015, pp. 497–517.

[10] S. Ghosh, J. Hiser, and J. W. Davidson, “Replacement attacks against vm-protected applications,” in Proceedings of the 8th International Conference on Virtual Execution Environments, VEE 2012, London, UK, March 3-4, 2012 (co-located with ASPLOS 2012), 2012, pp. 203–214.

[11] D. Kirovski and F. A. P. Petitcolas, “Replacement attack on arbitrary watermarking systems,” in Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers, 2002, pp. 177–189.

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

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

[14] F. Chevalier, D. D’Souza, R. M. Matteplackel, and P. Prabhakar, “Automata and logics over signals,” in Modern Applications of Automata Theory, 2012, pp. 555–584. [Online]. Available: https://doi.org/10.1142/9789814271059 0018

[15] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, Model Checking. Cambridge, MA, USA: MIT Press, 1999.

[16] 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.

[17] 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.Title Suppressed Due to Excessive Length

[18] 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.

[19] “Sony pictures hack.” [Online]. Available: https://en.wikipedia.org/wiki/Sony Pictures hack.

[20] J. Ming, Z. Xin, P. Lan, D. Wu, P. Liu, and B. Mao, “Impeding behavior-based malware analysis via replacement attacks to malware specifications,” J. Computer Virology and Hacking Techniques, vol. 13, no. 3, pp. 193–207, 2017.

[21] “Büchi automata and model checking.” [Online]. Available: http: //flolac.iis.sinica.edu.tw/flolac09/lib/exe/buechi_automata_4on1.pd

Publication from this project:

Debleena Das, Ansuman Banerjee, Sandeep K. Shukla, “An automata theoretic framework for detecting Schedulability Attacks on Cyber-physical Systems”, EAIT 2018: Fifth International Conference on Emerging Applications of Information Technology, 2018.