10-30-2016

A Method for Revealing and Addressing Security Vulnerabilities in Cyber-physical Systems by Modeling Malicious Agent Interactions with Formal Verification

Dean C. Wardell
Air Force Institute of Technology

Robert F. Mills
Air Force Institute of Technology

Gilbert L. Peterson
Air Force Institute of Technology

Mark E. Oxley
Air Force Institute of Technology

Recommended Citation

Wardell, D. C., Mills, R. F., Peterson, G. L., & Oxley, M. E. (2016). A Method for Revealing and Addressing Security Vulnerabilities in Cyber-physical Systems by Modeling Malicious Agent Interactions with Formal Verification. Procedia Computer Science, 95, 24–31.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 24 – 31

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

A Method for Revealing and Addressing Security Vulnerabilities in Cyber-Physical Systems by Modeling Malicious Agent Interactions with Formal Verification

Dean C. Wardell*, Robert F. Mills, Gilbert L. Peterson, Mark E. Oxley

Air Force Institute of Technology, 2950 Hobson Way, Dayton OH 45433

Abstract

Several cyber-attacks on the cyber-physical systems (CPS) that monitor and control critical infrastructure were publically announced over the last few years. Almost without exception, the proposed security solutions focus on preventing unauthorized access to the industrial control systems (ICS) at various levels – the defense in depth approach. While useful, it does not address the problem of making the systems more capable of responding to the malicious actions of an attacker once they have gained access to the system. The first step in making an ICS more resilient to an attacker is identifying the cyber security vulnerabilities the attacker can use during system design. This paper presents a method that reveals cyber security vulnerabilities in ICS through the formal modeling of the system and malicious agents. The inclusion of the malicious agent in the analysis of an existing systems identifies security vulnerabilities that are missed in traditional functional model checking.

Keywords: vulnerability detection; model checking; malicious agents; industrial control systems

* Corresponding author. Tel.: 1-321-961-5009
E-mail address: dean.wardell@us.af.mil

doi:10.1016/j.procs.2016.09.289

1. Introduction

Industrial control systems (ICS) have been "found to be rife with vulnerabilities"[1]. Researchers have put significant effort into developing security policies, practices and bolt-on security measures. The security measures have been primarily focused on intrusion detection, user authentication and countering malware. These multiple protective layers make up the 'Defense in Depth' strategy that is essential for system protection. Defense in Depth provides preventive measures, but more attention needs to be given to designing the systems to be more resilient to attacks if and when malicious actors do gain access.

A primary step in making these systems more resilient is identifying security vulnerabilities unique to the operational technology (OT) as opposed to those known to be in related IT systems. These vulnerabilities are usually inherent in the ICS design, but some are emergent from the combined system of systems that comprise so many ICS.

Once the vulnerabilities have been identified, steps can be taken to address them. The corrective actions can range from simple updates in the controller code, to redesigning portions of the system to add or change hardware. Depending on the criticality of the system, and the extent to which it is vulnerable to attack, an advanced fault tolerant or adaptive control strategy may be appropriate.

To begin the process of making a control system more resilient against a malicious agent that gains access to it, this paper proposes a novel application of formal system verification to identify cyber security vulnerabilities. This method is intended to be used as part of an overall risk management strategy and to assist those performing vulnerability analysis on ICS. We demonstrate this method on two different ICS controllers and then show how many of the vulnerabilities can be improved or even eliminated by making simple updates to the controller logic.

2. Related Work

Vulnerability assessments are used to identify where a system may be susceptible to attack. While primarily used on IT systems, cyber vulnerability assessments have also been advocated for use with ICS[2,3]. While useful, these are a manual processes requiring extensive system expertise.

In past work[4], the authors take steps to automate the vulnerability assessment of binary software programs (vs. source code) by enhancing traditional black-box fuzzing of the windows server with a genetic algorithm. A more recent example[5] uses fuzzing and an SMT solver to automate the vulnerability analysis of commercial binary programs. These programs have not been applied to ICS OT programs.

From a control theory perspective, research has been accomplished[6,7] in mathematically modeling control systems to better understand their functionality and improve their security. These models are used to produce simulations and conduct testing[8,9]. System testing is useful but suffers from the needle-in-a-haystack problem.

Unlike system testing, formal verification is a more complete solution. Formal verification of IT software and protocols has been well researched and some work has focused on modeling threats[10,11] and one specifically modeled and checked malicious interactions with the program[12].

The idea of formally verifying ICS through model checking has been explored to some extent since 2001. The research in this area has been focused on checking the functionality of the systems[13,14]. A lesser portion of the research has focused on verifying safety functions of control systems[15,16] and only recently have a few researchers considered model checking for security purposes[17].

Research in modeling human interactions with cyber-physical systems (CPS) has primarily considered correct user actions[18,19] with some consideration of the possibility of unintentional user errors[20,21]. None of these specifically model check malicious actions with the CPS.

It is at the intersection of 'model checking control systems for security' and 'modeling malicious agent interaction' that we focus our efforts.

Existing reference material[22,23,24] describes the different types of attacks that can executed against ICS and supervisory control and data acquisition (SCADA) systems. From these, the answer to the question 'what can a malicious agent do to the system once they gain access?' distils out to five general types of malicious interactions with a system:

Changing set points. Set points are used to set conditions under which the controller will start and/or stop particular functions. An authorized user/operator can change these set points by accessing the controller from an engineering/operator console. A malicious agent can change the set points in an effort to drive the system to an undesirable state.

Falsifying sensor input. The system controller relies on input from sensors to properly direct system operations. This attack can be accomplished by capturing a normal data packet transmission and then retransmitting that data to the controller when the actual sensor reading has changed or by simply blocking the sensor signal to the controller.

Sending harmful control signals. After gaining access to the system the attacker can monitor and learn the control signals, then send counterfeit signals or simply capture and retransmit control signals. A sophisticated agent could also send false feedback to the controller.

Changing Operator Display. Older systems frequently provide status data to the operator via feedback from system sensors and sub-controllers. In these systems the falsified sensor data (described above) would also provide that same false data to the operator. More modern systems use a digital human-machine interface (HMI). An HMI, can be remotely hacked and blanked out or possibly made to display false data.

*Rerouting or disrupting communications***.** Attacks that affect system communications are more common with SCADA and distributed control systems which rely heavily on communications protocols. For the examples in this paper, we focus on the first three types of attacks which are controller-centric.

3. Methodology

The input to the analysis is an accurate verification model of the system under assessment. This baseline model represents all functions of the controller, the physical components and the communications between them as a finite state machine. We identify all undesirable states for the system and include linear temporal logic (LTL) never-claims for each undesirable state.

After verifying the baseline system model, all system set points, inputs from sensors, feedback from components and output commands from the controller are identified. With this information, we model the malicious actions (attacks) against the system. The possible malicious actions are:

- Changing system set points
- Preventing a sensor from providing input to the controller
- Causing a sensor to provide out-of-bounds input
- Causing a sensor to repeatedly provide the same false input
- Sending harmful command signals to components
- Sending harmful command signals and falsifying component feedback

Finally, we verify the system model, complemented with the malicious agent models, and allow the model checker to identify the malicious agent actions that cause the system to enter an undesirable state.

4. Applications

The method described is applied to two CPS; a nuclear power plant steam boiler design and an existing commercial heating, ventilation and air conditioning (HVAC) control system. We use PROMELA to model the systems with the physical plant and the controller each modeled as separate processes which run concurrently in PROMELA. Likewise, each of the malicious agents is modeled as a separate process.

4.1. Steam Boiler System

The Steam-boiler Control Specification Problem[25] is based on a specification by the “Institute for Risk Research” and the “Protection and Nuclear Safety Institute” and is very formal.

The task of the system control is to maintain the water level in the boiler between two pre-set limits. Overfilling the boiler or allowing it to run dry while heat is applied could damage the system and are unacceptable conditions. As with all controllers, a deadlock or live lock condition is also unacceptable.

Because safety and reliability are important for this system, the controller has fault tolerant features designed into it. The controller can switch between five different modes[25]. Two are normal operational modes (*Initialization* and *Normal),* two are special modes the controller switches to when a fault is detected (*Degraded* and *Rescue*) and the fifth is a controlled shutdown (*Emergency Stop*). The controller invokes the *Emergency Stop* mode when it determines it can no longer safely control the boiler system.

The baseline model used in this paper was published by Siegfried Löffler[26]. We selected this model because it fully satisfies the requirements of the original control problem, effectively implements the various operational modes of the controller, and is efficiently designed so as to not require excessive amounts of time to verify.

The adjustable set points for the system are described in Table 1.

Table 1. Set Points for the Steam Boiler Controller.

Set Points	Description	Normal Value (liters)
M1	Minimum allowable water level	100
N1	Normal low water level	400
N2	Normal High water level	600
M2	Maximum allowable water level	900

The inputs to the controller are the water level (level) and the status (on/off) of each of the four pumps. The controller acts on a drain valve (open/close) and the four filling pumps (on/off).

We modeled three types of malicious actions against the baseline system: Changing set points (normal, min and max water levels), falsifying sensor inputs to the controller (water level and pump status) and sending harmful control signals (to the pumps and the valve).

Each of these malicious interactions represents an attacker trying to drive the system to an undesirable state. The researcher building the agent models need not know a priori exactly how the system will respond to a given attack. Like a real attacker he/she may try various inputs to see which are successful.

Even with the safety and fault tolerant features of the controller design, we found it vulnerable to several different cyber-attacks. Those modeled attacks to which we found this system is vulnerable are described in Table 2.

Table 2. Summarized Results of Malicious Agent Interaction with Baseline and Updated Steam Boiler Systems.

Description	Baseline Results	Updated Results
Changing Set Points		
M1 and N1 = -100	Dry Boiler	Normal Operation
M2 and N2 = 1400	Overfill	Normal Operation
Falsified Sensor Input		
False level < 400 in Init mode	Overfill	Emergency Stop
False 400 < level < 600 in Init mode	Dry Boiler	Rescue Mode
False 600< level in Init Mode	Ctrl Deadlock	Emergency Stop
False 200< level < 400 in Normal mode	Overfill	Rescue Mode
False 400 < level < 600 in Normal mode (pumps on)	Overfill	Rescue Mode
False 400 < level < 600 in Normal mode (pumps off)	Dry Boiler	Rescue Mode
False 600 < level < 800 in Normal mode	Dry Boiler	Rescue Mode
Harmful Command Signals		
Valve held open in Init mode	Ctrl Deadlock	Emergency Stop
Valve held closed in Init mode	Ctrl Deadlock	Emergency Stop

The safety and fault tolerant features that are part of the controller design do help in handling attempted attacks against the pumps. The fact that the system design uses multiple (redundant) pumps to fill the boiler also eases the challenge of countering a failure of, or attack on, some of the pumps.

We addressed the vulnerabilities to the *changing set points* attacks by adding simple guard statements that ensure the set points are within pre-set bounds before using them. The *falsified sensor input* and *harmful command signal* attacks were addressed by logic checks to the controller that simply compared the expected sensor input against the reported values. When the difference between these two reached a certain threshold, the *emergency stop* routine was invoked. Because a controlled shutdown is an acceptable solution for this particular system, this was the solution chosen for this simple demonstration.

After updating the model we rechecked the system with each of the agent models and found the controller was now able to function in one of its designed modes in every case. The worst results we found were those in which the controller executed a controlled shutdown (Table 2, third column).

4.2. Commercial HVAC System

We also applied the method to a commercial HVAC system using the Siemens APOGEE® controller. Currently, the U.S. Air Force is using this system to automate its building HVAC systems on several bases[27].

In this particular example, the HVAC system is designed to keep the Combat Air Operations Center (CAOC) server room temperature within a range of temperatures set by the users. The only unacceptable condition for this system is allowing the server room temperature to rise above 90^o. For this specific application there are six set points of interest (Table 3).

Table 3. Set Points of Interest in APOGEE Controller.

Set Point #	Description	Normal Value
6	High temp setting (user defined)	70^o
7	Low temp setting (user defined)	60^o
11	Minimum allowable temp	55^o
12	Maximum allowable temp	80^o
86	Minimum time between mode switches	10 minutes
90	Variation from pts 6 or 7 to trigger action	1^o

The controller takes the room temperature as input and sends ‘on’ or ‘off’ commands to the air conditioner, the heater and the fan. It also sends a temperature display and colored status lights to the HMI.

The controller utilizes only minimal safety features to mitigate the impact of a malfunction or user error. If the user inputs an upper temperature (set point 6) higher than set point 12, the system uses set point 12. Likewise, if the user inputs a low temperature (set point 7) lower than set point 11, the system uses set point 11.

If the controller senses a temperature sensor malfunction (no input) it uses the last reported temperature.

The model being checked was developed based on the documented operational data[28] for the APOGEE® controller and data provided by Siemens.

The HVAC system modeled here has no redundant components and the controller was not designed to be fault tolerant or adaptive. Since the purpose of this system is keeping a server room cooled, an emergency shutdown is not an acceptable solution to a fault or attack. We, therefore, focus our attention on how much (if any) warning the system can provide to the operator in the event of a failure or attack in this example problem.

The finite state machine (Table 4) we used to represent the controller has four states. States 1-3 are normal operation states, while state 4 is an abnormal operation state which triggers a warning light to operator.

- S1 – Heater = ON, Fan = ON, A/C = OFF, Mode = HEAT
- S2 – Heater = OFF, Fan = OFF, A/C = OFF, Mode = (not specified)
- S3 – Heater = OFF, Fan = ON, A/C = ON, Mode = COOL
- S4 – Heater = OFF, Fan = ON, A/C = ON, Mode = COOL

The inputs to the FMS are the room temperature (RT) and the time elapsed (Time) since the last mode change (if applicable). The outputs of interest are the system status lights and reported temperature displayed at the HMI.

Table 4. Transition Table for the HVAC Controller Finite State Machine.

Input		States		Output Display	
RT	Time	Current	Next	Temp	Status Light
≥ 61° and < 71°	~	S1	S2	61° – 70°	Green
≥ 71° and < 80°	≥10 Min	S2	S3	71° – 79°	Green
≥ 80°	~	S3	S4	≥80°	Yellow
≥ 69° and < 80°	~	S4	S3	69° – 79°	Green
≥ 59° and < 69°	~	S3	S2	59° – 68°	Green
< 59°	≥10 Min	S2	S1	< 59°	Green

We modeled three types of malicious actions against the baseline system: Changing set points, falsifying sensor inputs to the controller (temperature sensor) and sending harmful commands to the various system components (A/C, heater and fan).

Each of these malicious interactions represents an attacker trying to cause the server room to overheat. Those attacks that resulted in the room overheating are shown in the first three columns of Table 5 along with the maximum temperature displayed to the operators as well as the temperature at which the system status light turned yellow (if ever) to warn the operators of abnormal operations.

Table 5. Summarized Results of Malicious Agent Interactions with the Original and Updated HVAC Control System.

Malicious Agent	Original Model		Updated Model		
Description	High Temp	Yellow Lt.	Result	High Temp	Yellow Lt.
Changing Set Points					
sp7 = 90° (in Heat mode)	>90°	80°	Normal Ops	71°	N/A
sp12 = 95° and sp6 = 90°	>90°	never	Modified Ops	86°	85°
sp12 = 95° and sp7 = 90°	>90°	never	Normal Ops	71°	N/A
sp90 = 25°	>90°	80°	Normal Ops	71°	N/A
sp86 = 60 minutes	>90°	80°	Normal Ops	71°	N/A
Falsified Sensor Input					
False room temp = 70° (with A/C is off)	70°	never	Modified Ops	70°	72°
No room temp input (with A/C is off)	70°	never	Modified Ops	70°	72°
Harmful Command Signals					
A/C Off in Cool mode	>90°	80°	Overheated	>90°	74°
Heat On in Cool mode	>90°	80°	Overheated	>90°	74°
Fan Off in Cool mode	>90°	80°	Overheated	>90°	74°

Of particular concern are those attacks in which the agent falsified the input from the room temperature sensor and allowed the room to overheat but the system gave NO warning to the operators. An Air Force "Red Team" successfully carried out this same attack against a real-world system during a recent Red Flag[27] exercise.

Since a controlled shutdown is not a good operational solution for this system, the changes made to address these vulnerabilities focused on providing the users with more advanced warning of a problem (last three columns of Table 5). By adding guard statements to the controller input, we were able to prevent the *Altered Set Point* from driving the system to an unacceptable state. As in the previous example, checks for actual vs. expected system behavior and some simple adaptive behavior prevented the *Falsified Sensor Input* agents from causing the server room to overheat. The lack of redundant actuators in the system limits the options for countering attacks on the A/C, heater and fan, but with simple updated logic checks we were able provide more advanced warning to the operators in the cases when the agent was sending harmful control signals.

5. Conclusion

We have presented a novel method for identifying cyber security vulnerabilities in industrial control systems. We accomplish this by complementing ICS systems (modeled as finite state machines) with models of malicious agent interactions and formally verifying the combined models. The output from the model checker is a path to the vulnerability. Any corrective action taken can likewise be verified by model checking. This method provides a simple means for system engineers, designers and/or operators to identify and address security vulnerabilities in their planned or existing CPS before they are actually subjected to a real-world attack.

The focus of this paper is on the identification of security vulnerabilities. As such, the sample solutions provided in these examples are very simple. The next step of this research is to use the vulnerability detection capability as a basis developing more capable and adaptable controllers for ICS.

References

1 Zetter, Kim, *A Cyberattack Has Caused Confirmed Physical Damage for the Second Time Ever*, Wired Online, 8 January 2015. http://www.wired.com/2015/01/german-steel-mill-hack-destruction

2 Baybutt, P., An Asset-Based Approach for Industrial Cyber Security Vulnerability Analysis, *Process Safety Progress*, Vol. 22, No. 4, pages 220-292, 2003.

3 Sarwate, A., *2015 Industrial Control System Vulnerability Trends*, Presentation at RSA Conference 2015, Session SEC-F04, July 2015.

4 Sparks, S., Embleton, S., Cunningham, R. and Zou, C., Automated Vulnerability Analysis: Leveraging Control Flow for Evolutionary Input Crafting, *Proceedings of the IEEE 23rd Annual Computer Security Applications Conference*, pages 477-486, 2007.

5 Kimball, W., *A Formal Approach to Vulnerability Discovery in Binary Programs*, PhD Dissertation, Air Force Institute of Technology, 2013. AFIT -ENG-DS-13-J-03

6 Cardenas, A.A., Roosta, T. and Sastry, S., Rethinking security properties, threat models, and the design space in sensor networks: a case study in SCADA systems, *Ad Hoc Networks* 7, 1434-1447 (2009)

7 Burmester, M., Magkos, E. and Chrissikopoulos, V., Modeling security in cyber-physical systems, *International Journal of Critical Infrastructure Protection* 5, 118-126 (2012)

8 Genge, B., Siaterlis, C., Fovino, I. and Masera, M., A cyber-physical experimentation environment for the security analysis of networked industrial control systems. *Computers and Electrical Engineering* 38, 1146-1161 (2012)

9 Combita, J. F., Giraldo, J., Cardenas, A. A. and Quijano, N., Response and Reconfiguration of Cyber-Physical Control Systems: A survey. *IEEE 2nd Colombian Conference on Automatic Control* (CCAC), 1-6 (2015)

10 Sheyner, O., et al, Automated Generation and Analysis of Attack Graphs, *Proceedings of the 2002 IEEE Symposium on Security and Privacy* (S&P'02), 2002.

11 Bau, J. and Mitchell, J. C., Security Modeling and Analysis, *IEEE Security & Privacy* 9, (3), pages 18-25, 2011. DOI: 10.1109/MSP.2011.2

12 Yu, W. Y., et al, Modeling and Verification of Online Shopping Business Processes by Considering Malicious Behavior Patterns, *IEEE Transactions on Automation Science And Engineering*, 2014.

13 Fernandez, B., Blanco, E. and Merezhin, A., Testing & Verification of PLC Code for Process Control, *Proceedings of ICALEPCS2013*, 2013.

14 Carpanzano, L., et al, Automated Formal Verification for Flexible Manufacturing Systems, *Journal of Intelligent Manufacturing*, October 2014, Volume 25, Issue 5, pp 1181-1195.

15 Valkonen, J., et al, Formal Verification of Safety Automation Logic Designs, *Automaatio XVIII Seminaari 2009*, 17-18.3.2009

16 Bartha, T., et al, Verification of an Industrial Safety Function Using Coloured Petri Nets and Model Checking, *Proceedings of the 14th International Conference on Modern Information Technology in the Innovation Processes of the Industrial Enterprises*, 2012. pp 472-485.

17 Armstrong, R.C., Punnoose, R. J., Wong, M. H. and Mayo, J.R., *Survey of Existing Tools for Formal Verification*, SANDIA REPORT SAND2014-20533, Unlimited Release, Printed December 2014

18 Combefis, S., Giannakopoulou, D.and Pecheur, C., State Event Models for the Formal Analysis of Human-Machine Interactions, *Formal Verification and Modeling in Human-Machine Systems*: Papers from the AAAI Spring Symposium, 2014

19 Billman, D., Work Representations for Evaluating and Modeling Human-Machine Systems, *Formal Verification and Modeling in Human-Machine Systems*: Papers from the AAAI Spring Symposium, 2014

20 Bolton, M. and Bass, E., Using Task Analytic Models and Phenotypes of Erroneous Human Behavior to Discover System Failures Using Model Checking, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 2010 54: 992

21 Javaux, D., et al, A Methodology for Analyzing Human-Automation Interactions in Flight Operations Using Formal Verification Techniques, *Formal Verification and Modeling in Human-Machine Systems*: Papers from the AAAI Spring Symposium, 2014

22 Igure, V.M., Laughter, S. A. and Williams, R. D., Security issues in SCADA networks, *Computers & Security*, 25:498-506, 2006

23 Caswell, J., *A Survey of Industrial Control System Security*, Washington State University, St Louis, MO., 2011. http://www.cse.wustl.edu/~jain/cse571-11/ftp/ics/index.html

24 Zhu, B., Joseph, A. and Sastry, S., A Taxonomy of Cyber Attacks on SCADA Systems, *Proceedings of the International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 380-388, 2011.

25 Abrial, J-R., *Steam-boiler control specification problem*, http://www.informatik.unikiel.de/~procos/dag9523/steam-boiler-problem.ps.Z, 1994

26 Löffler, S., *From Specification to Implementation: A PROMELA to C Compiler*, Ecole Nationale Supérieure des Télécommunications, Département Réseaux, Paris, France, 1998.

27 Air Force Institute of Technology, *Red Team Playbook*, *Industrial Control System Integration*, Red Flag 15-3, 2015.
28 Siemens, *APOGEE Actuating Terminal Equipment Controller – Electronic Output Owner's Manual*, Building Technologies, 125-3209, Rev 1, March 2004, section 3, pages 67–80.