The Use of Runtime Verification for Identifying and Responding to Cybersecurity Threats Posed to State Actors During Cyberwarfare

Jeremy Straub
Institute for Cyber Security Education and Research
North Dakota State University
Fargo, ND, USA
jeremy.straub@ndsu.edu

Abstract— **This paper considers the utility of the use of runtime verification techniques for detecting and responding to cybersecurity threats. To this end, it considers two questions: First, it evaluates the efficacy of runtime verification for identifying zero-day threats and threats that are not otherwise widely known based up system operations. Second, it considers the particular use of these techniques by state actors (i.e., nation states engaged in declared or undeclared cyberwarfare) who are likely to encounter a greater level of such vulnerability exploits than individuals or private businesses during the normal operations. Drawing on the analysis in the two foregoing areas, the paper concludes by identifying key areas of needed future work to support runtime verification's application in this area.**

Keywords—runtime verification, cybersecurity, threats, cyberattacks, cyberwarfare

I. Introduction

Information technologies are an integral part of the everyday world and have created a new virtual battlefield. While cyberwar may happen in a virtual space, the consequences of it can often be felt in the real world. This includes consequences such as the theft of individuals personal information [1]–[3] and more pronounced impacts if cyber-physical systems – physical hardware under computer control – are targeted and successfully breached [4].

Cyberattacks and cyberwarfare can be used for a variety of purposes. In some cases the goal of the attack is to compromise specific systems for a targeted immediate goal [5]. In many other cases, there is a larger goal related to positioning an attacker (whether a nation state, criminal organization, terrorist or individual attacker) to wield influence over the attacked party [6], [7]. One key tool that sophisticated attackers have in their arsenal is so-called 'zero-day' attacks [8]. These never-before-used attacks, along with attacks that are still new enough that software vendors may not have completed or distributed fixes, can potentially be deployed against government and military systems. However, attackers may also target civilian systems that provide key architecture, such as SCADA systems [4], the banking sector [9] and transportation [10], [11]. Traditional attach signature-based models [12] are unable to effectively detect or initiate a response against these inherently pre-signature-availability attacks.

This paper builds on prior work in runtime verification [13] and its application to anomaly detection [14] and cybersecurity [15] to propose a paradigm to aid in solving this problem. Specifically, a methodology for both the manual and autonomous development and refinement of models for anomaly-based incident detection is proposed. Further, its efficacy for use by sophisticated state actors and those targeted by them is discussed.

II. Background

This section discusses prior work which provides a foundation for the work presented in this paper. First, cybersecurity threats and escalation are discussed. Second, a brief discussion of runtime verification is presented. Finally, prior work on runtime verification for cybersecurity purposes is considered.

A. Cybersecurity Threats and their Escalation

A wide variety of cybersecurity threats exist across numerous sectors. These range from threats targeting private individuals to directly or indirectly capture their personal information [2], [3], attacks targeting national security and intelligence capabilities [16] to attacks targeting cyber-physical systems that directly interact with the environment and humans [17]. This last group represents a more immediate threat, as a compromised cyber-physical system could prospectively injure or kill a human bystander or proximal worker before the compromise is detected. Given this, thinking of data as the principal target of attack (e.g., [18]) fails to fully consider the scope of the problem. Jang-Jaccard and Nepal [19] note that "the development of more innovative and effective malware defense mechanisms has been regarded as an urgent requirement" due to the escalating threat.

B. Runtime Verification

Significant prior work exists related to runtime verification [13], [20] that serves as a foundation for the current work. Prior techniques have been proposed which include those based on and using rules [21], state estimation [22] and predictive analysis [23]. An event-triggered approach, which analyzed event traces, was proposed by d'Amorim and Havelund [24]. Falcone, Fernandez and Mounier [25] proposed a model that was based on the identification of verifiable properties for "safety-process classification".

In addition to the foregoing, a number of frameworks (e.g., [26]) have been proposed for various applications. A number of techniques considering linear temporal logic [27], [28] and trimmed lineartime temporal logic [28] have also been proposed.

C. *Prior Work on Runtime Verification for Cybersecurity*

Prior work has also focused, specifically, on the use of runtime verification techniques for cybersecurity purposes [15]. Applications have ranged from unmanned aerial systems safety [29] and safety monitoring for embedded systems [30] to protecting "combat systems" [31]. A wide variety of techniques have been proposed including those using side-channel fingerprinting [32], power profiles [33], temporal logic [34], metamorphic testing [35], code mutation [36], chaos theory [15], behavior and game theories [37] and return-oriented programming [38], among others.

Robots [39], including UAVs [29], [40] and programmable logic controllers [41] have been among the numerous applications. These applications stand out, from a security perspective, due to the prospective threat that they represent to the public, if they are compromised.

III. Runtime Verification for Zero-Day Detection

The challenge presented by 'zero-day' vulnerability exploitation is that a detection and response strategy cannot presume that the attack or its particular outcomes (including data theft, system maloperation or other results) have been seen before. In fact, the most problematic attack to defend against would incorporate a new attack vector that targeted a previously unexploited vulnerability and had either a new maloperation outcome or used a new data exfiltration method. Under these circumstances, the attack may not trigger any pre-existing attack signature-based intrusion detection tool.

This section presents an approach, thus, based on modeling the existing system. It begins with a discussion of how a model can be developed (either autonomously or with manual input) and how the system can be characterized using this model. Then, the use of this model for attack detection is discussed.

A. *Development of a Model of the System*

The development of a model, inherently, involves characterizing the performance of the system. Attack signature-based models seek to characterize how an attack behaves or how a system behaves when subjected to a particular type of attack. This approach benefits from being somewhat generic, as an attack signature can be captured on one system and prospectively used to detect similar attacks against a wide variety of systems. However, as previously discussed, this approach cannot be used against unknown and previously uncharacterized attacks.

Fundamentally, thus, the goal of a model for responding to new attacks must be to characterize the normal and acceptable operations of the system. When the bounds of normal and acceptable operations are known, anything outside of these bounds is an anomaly that represents a potential threat which should be investigated and potentially responded to.

The first key question, in this process, is to determine what to characterize the system in terms of – that is, what to monitor and measure. Two different approaches to this are proposed, under the first, a system administrator uses her or his experience to identify relevant metrics. The system collects data related to these metrics and uses this to create and iteratively refine the model of the system. Figure 1 depicts this process.

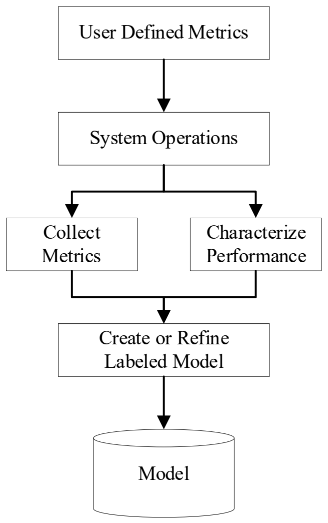


Fig. 1. Model development from user-defined metrics.

However, this may not always be possible. Additionally, system administrators may not be aware of all metrics that are relevant to system characterization. For this reason, an alternate approach, based on automated detection is proposed (and depicted in Figure 2). Under this model, all of the metrics that can be monitored by the monitoring toolkit are considered. This process begins with an initial run to determine what metrics are showing variance and appear relevant to characterizing system performance. Once these metrics are identified, the characterization system collects data on them and uses this for system characterization. It is important to note that, typically, the system will need to be characterized under several different modes of operation. Relevant metrics should be identified based on all of these modes and, in particular, metrics which are different under different modes of operations should be identified.

Typically, the system will need to be run under normal, optimal and degraded conditions. While the goal is not to characterize the system under compromised conditions (as this would be an attack signature-based approach), running the system under simulated compromised conditions may also be helpful to identify metrics that are demonstrably different under these conditions. Of course, different types of compromise may have dramatically different impacts and affect different metrics. Thus, while some simulation of compromise may be useful for metric identification, focus should be on characterizing the correctly-operating system in terms of relevant metrics, not trying to identify metrics of compromise.

Metrics that are selected, whether autonomously or manually identified, should have several key characteristics. They must be observable with minimal impact to system operations (particularly for metrics that will be used continuously to detect symptoms for further escalation). Metrics that characterize performance, as well as those that characterize outputs and outcomes should both be selected.

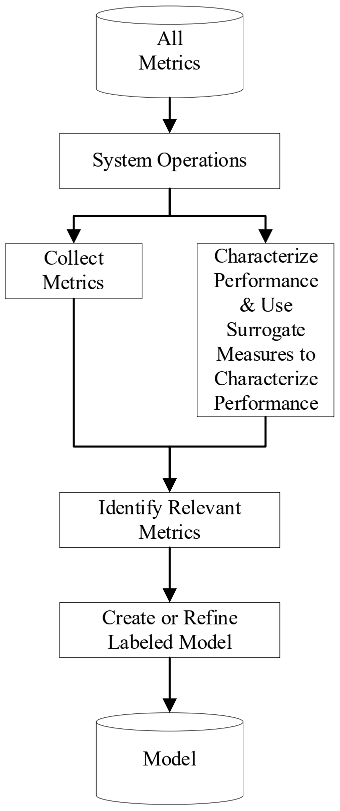

All Metrics
System Operations
Collect Metrics
Characterize Performance & Use Surrogate Measures to Characterize Performance
Identify Relevant Metrics
Create or Refine Labeled Model
Model

Fig. 2. Model development with autonomous metric identification.

The metrics that are selected should fall into several different groups. The factors that are used for this classification are the difficulty and cost of collection, whether a metric is a leading indicator of a problem, the percentage of problems that it serves as an indicator-metric for and whether it is a metric for an issue or characteristic of particular importance. This is depicted in Figure 3. The resulting groups are summarized in Figure 4.

Level 1 metrics should include those that are general purpose indicators of potential issues (i.e., those that broadly characterize proper system operations or are likely to change if system operations are impaired). Level 1 also includes metrics that identify specific issues that are not covered by these general metrics (level 1S metrics) and those that identify critically important issues that should be continuously monitored for (level 1I metrics). Ideally, all level 1 metrics will be leading indicators of problems, allowing response before the problems escalate. They should also be as low cost (in terms of performance impact and other costs) as possible as they will be run continuously.

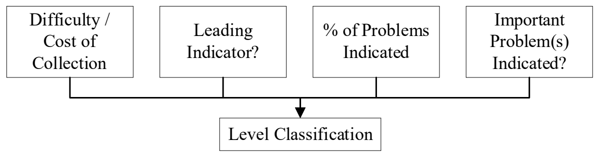
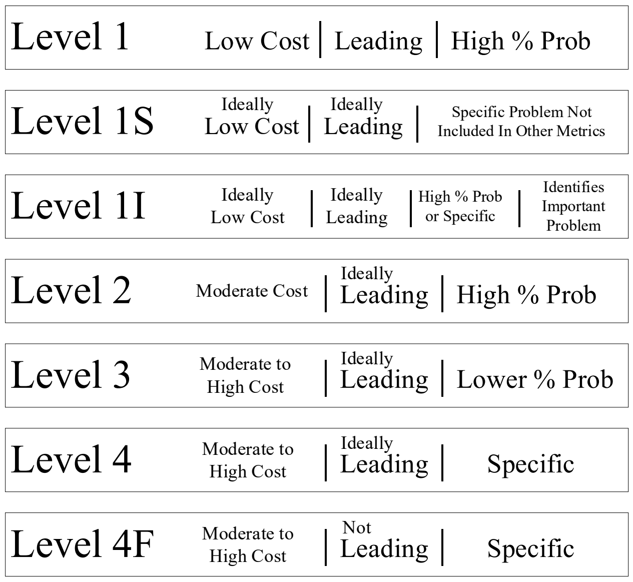

Difficulty / Cost of Collection
Leading Indicator?
% of Problems Indicated
Important Problem(s) Indicated?
Level Classification

Fig. 3. Inputs to level classification.



Level 1	Low Cost	Leading	High % Prob	
Level 1S	Ideally Low Cost	Ideally Leading	Specific Problem Not Included In Other Metrics	
Level 1I	Ideally Low Cost	Ideally Leading	High % Prob or Specific	Identifies Important Problem
Level 2	Moderate Cost	Ideally Leading	High % Prob	
Level 3	Moderate to High Cost	Ideally Leading	Lower % Prob	
Level 4	Moderate to High Cost	Ideally Leading	Specific	
Level 4F	Moderate to High Cost	Not Leading	Specific	

Fig. 4. Classification levels.

Level 2 metrics are those that have a higher cost and may refine the understanding of the current system status provided by the level 1 metrics. Level 3 metrics may be even higher cost and should be more specific to begin to isolate the particular causes of the detected issue or issues. Finally, level 4 metrics should be specific to allow the particular anomalous system operations to be isolated. Because of the specificity of these (and the potential need to run numerous level 4 metrics, selected based on the narrowing of the issue by the level 3 metrics) the cost of these is expected to be higher than other metrics. Finally, level 4F metrics collect forensic data about what is or has happening and are not expected to be a leading indicator.

The different classifications of metrics determine how, when and how frequently they are used by the monitoring system. The use of these metrics and the comparative use of metrics of different classifications is discussed in the subsequent section.

B. Measurement and Comparison to Model

Once metrics are identified and the system is characterized in terms of them, the system being secured / verified must be monitored using the metrics. Figure 5 depicts the relationship between the two systems. While the monitoring can, prospectively, be conducted on the primary system, this is not ideal as an attack may potentially impair both the primary system and the incident detection / verification system. Under the ideal (two system) configuration, the metrics are requested by the verifying system and returned to a database in near real time. The verification system then processes this data on a near real time basis. Notably, it is ideal that the verification system be appropriately resourced to keep up with this data flow. Otherwise, it will either lag behind real time, delaying issue detection. It may also need to skip over data to maintain temporal currency, which may result in attacks and issues being missed over.

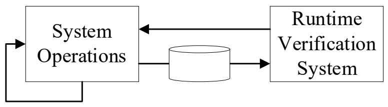


Fig. 5. Interaction between system operations and runtime verification module.

The overall model for the operations of the verification system is presented in Figure 6. Specifically, the proposed approach utilizes a model of escalation. The level 1 processes (including L1, L1S and L1I) are run continuously. If a level 1 process metric shows an anomaly, then the level 2 metrics are collected and evaluated (if level 2 metrics can be collected continuously without significant cost, the metrics may be continuously collected but only processed after triggering based on a level 1 metric). A similar approach is taken with the level 3 and level 4 processes. Both are triggered based on an issue being detected at the immediately lower level (level 2 and 3, respectively).

Finally, if a level 4 process detects a problem, via one or more of its metrics, then the level 4 forensics (level 4F) processes are triggered to collect data. Additionally, administrator notification and other system-specific processes are triggered, based on the strong likelihood of system compromise (or other similarly problematic anomalous situation) having occurred.

IV. Nation States and Cyber Warfare

With a paradigm for identifying ‘zero-day’ and other unknown-to-operator attacks proposed, a question exists as to its efficacy for the issues of nation states engaged in cyberwarfare and those engaged in cyberwarfare against (or targeted by) them. A key question, thus, is what are the defensive and other related needs of nation states engaging in cyberwarfare and those targeted by them?

This is a complex question that cannot, for purposes of brevity, be fully explored here. However, these needs can be classified into two key areas. The first is the need to defend systems. To this end, the proposed system must be able to identify problems quickly enough to provide an opportunity to respond to prevent or mitigate the impact of adversaries’ attacks. The second need is the need for attribution to allow retaliatory action via cyber or other means. The ability to attribute attacks and retaliate is integral to deterrence [42], as this is the primary inventive for an adversary to not attack in order to achieve whatever aims it would otherwise desire. To meet this second need, it is critical that the system collect data (as part of the level 4F processes and at other levels) that contains attack source data. It is also critical that this data be protected from loss or alteration. This is a key justification for the two-system model shown in Figure 5.

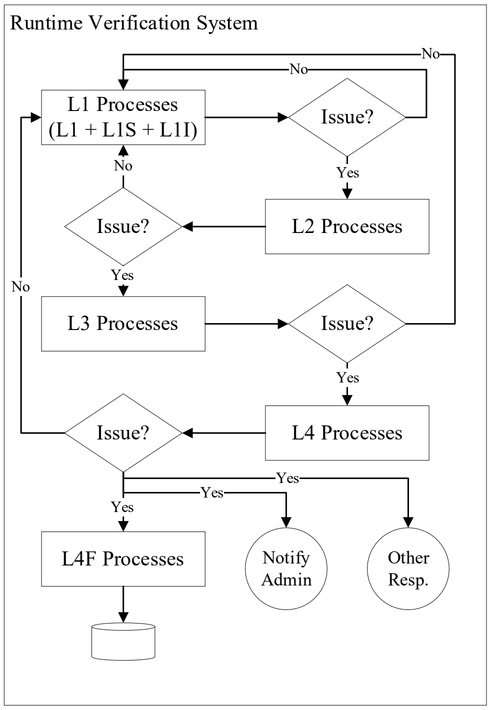


Fig. 6. Operations of the runtime verification module.

V. Conclusions and Future Work

This paper has presented a paradigm for the use of runtime verification as part of a detection and response system for ‘zero-day’ and other unknown-to-operator attacks. It has, in particular, focused on the efficacy of this model for use by nation states and those who may be targeted by them. The paradigm has been presented and explained and the different levels of processes required for it have been discussed. The utility of the system for meeting the needs of nation states have bene considered.

Significant work remains to be done in this area, in the future. In particular, additional work on the development and evaluation of a system to demonstrate and assess the proposed paradigm is required.

References

[1] N. Perlroth, A. Tsang, and A. Satariano, “Marriott Hacking Exposes Data of Up to 500 Million Guests - The New York Times,” New York Times, 2018. [Online]. Available: https://www.nytimes.com/2018/11/30/business/marriott-data-breach.html. [Accessed: 11-Dec-2018].

[2] T. Armerding, “The 18 biggest data breaches of the 21st century,” CSO Magazine, 20-Dec-2018.

[3] D. Winder, “Data Breaches Expose 4.1 Billion Records In First Six Months Of 2019,” Forbes, 20-Aug-2019.

[4] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke, “SCADA security in the light of Cyber-Warfare,” Comput. Secur., vol. 31, no. 4, pp. 418–436, 2012.

[5] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” IEEE Secur. Priv., vol. 9, no. 3, pp. 49–51, May 2011.

[6] J. Straub and T. Traylor, “Towards an Influence Model for Cybersecurity and Information Warfare,” in Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence, 2018.

[7] J. Straub and T. Traylor, “Introduction of a maritime model for cyber and information warfare,” in Proceedings - 2018 International Conference on Computational Science and Computational Intelligence, CSCI 2018, 2018, pp. 25–29.

[8] A. Kuehn and M. Mueller, “Shifts in the Cybersecurity Paradigm: Zero-Day Exploits, Discourse, and Emerging Institutions,” in Proceedings of the 2014 workshop on New Security Paradigms Workshop, 2014, pp. 63–68.

[9] A. L. Johnson, “Cybersecurity for Financial Institutions: The Integral Role of Information Sharing in Cyber Attack Mitigation,” North Carolina Bank. Inst., vol. 20, 2016.

[10] M. H. Eiza and Q. Ni, “Driving with Sharks: Rethinking Connected Vehicles with Vehicle Cybersecurity,” IEEE Veh. Technol. Mag., vol. 12, no. 2, pp. 45–51, Jun. 2017.

[11] G. C. Kessler, P. Craiger, and J. C. Haass, “A Taxonomy Framework for Maritime Cybersecurity: A Demonstration Using the Automatic Identification System,” TransNav, Int. J. Mar. Navig. Saf. Sea Transp., vol. 12, no. 3, pp. 429–437, Nov. 2018.

[12] A. M. Cansian, A. R. A. Da Silva, and M. De Souza, “An attack signature model to computer security intrusion detection,” in Proceedings - IEEE Military Communications Conference MILCOM, 2002, vol. 2, pp. 1368–1373.

[13] M. Leucker and C. Schallhart, “A brief account of runtime verification,” J. Log. Algebr. Program., vol. 78, pp. 293–303, 2009.

[14] O. S. Pieczul, “Detecting Anomalous Events Through Runtime Verification of Software Execution Using a Behavioral Model,” US 10,152,596 B2, 11-Dec-2018.

[15] H. Zhao, K. Kwiat, C. Kamhoua, and M. Rodriguez, “Applying Chaos Theory for Runtime Hardware Trojan Detection,” in Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2015.

[16] 114th Congress Committee on Oversight and Government Reform of the U.S. House of Representatives, “The OPM Data Breach: How the Government Jeopardized Our National Security for More than a Generation,” 2016.

[17] V. Bolbot, G. Theotokatos, L. M. Bujorianu, E. Boulougouris, and D. Vassalos, “Vulnerabilities and safety assurance methods in Cyber-Physical Systems: A comprehensive review,” Reliability Engineering and System Safety, vol. 182. Elsevier Ltd, pp. 179–193, 01-Feb-2019.

[18] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, “Data-Driven Cybersecurity Incident Prediction: A Survey,” IEEE Commun. Surv. Tutorials, vol. 21, no. 2, 2019.

[19] J. Jang-Jaccard and S. Nepal, “A survey of emerging threats in cybersecurity,” in Journal of Computer and System Sciences, 2014, vol. 80, no. 5, pp. 973–993.

[20] E. Bartocci, Y. Falcone, A. Francalanza, and G. Reger, “Introduction to Runtime Verification,” Lect. Notes Comput. Sci., vol. 10457, pp. 1–33, 2018.

[21] H. Barringer, A. Goldberg, K. Havelund, and K. Sen, “Rule-Based Runtime Verification,” in Proceedings of the International Workshop on Verification, Model Checking, and Abstract Interpretation, 2004, pp. 44–57.

[22] S. D. Stoller et al., “Runtime Verification with State Estimation,” in Proceedings of the International Conference on Runtime Verification , 2011, pp. 193–207.

[23] E. Bartocci et al., “Adaptive Runtime Verification,” in Proceedings of the InInternational Conference on Runtime Verification, 2012, pp. 168–182.

[24] M. d’Amorim and K. Havelund, “Event-Based Runtime Verification of Java Programs,” in Proceedings of the Workshop on Dynamic Analysis, 2005.

[25] Y. Falcone, J.-C. Fernandez, and L. Mounier, “Runtime Verification of Safety-Progress Properties,” in Proceedings of the International Workshop on Runtime Verification, 2009, pp. 40–59.

[26] F. Chen and G. Rosu, “MOP: An Efficient and Generic Runtime Verification Framework *,” in Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications, 2007, pp. 569–588.

[27] G. Rosu and K. Havelund, “Rewriting-Based Techniques for Runtime Verification,” Autom. Softw. Eng., vol. 12, pp. 151–197, 2005.

[28] A. Bauer, M. Leucker, and C. Schallhart, “Runtime verification for LTL and TLTL,” ACM Trans. Softw. Eng. Methodol, vol. 20, no. 14, 2011.

[29] J. Schumann, P. Moosbrugger, and K. Y. Rozier, “R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems,” Runtime Verification. p. 15, 2015.

[30] S. Gautham, G. Bakirtzis, M. T. Leccadito, R. H. Klenke, and C. R. Elks, “A Multilevel Cybersecurity and Safety Monitor for Embedded Cyber-Physical Systems,” in Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, 2019.

[31] J. Hamilton, “A High-Integrity Cybersecurity Framework for Combat Systems,” in Proceedings of the European Conference on Cyber Warfare and Security, 2017, pp. 157–164.

[32] S. Yang, A. Alaql, T. Hoque, and Swarup Bhunia, “Runtime Integrity Verification in Cyber-physical Systems using Side-Channel Fingerprint,” in Proceedings of the 2019 IEEE International Conference on Consumer Electronics, 2019.

[33] M. F. Bin Abbas, A. Prakash, and T. Srikanthan, “Power Profile based Runtime Anomaly Detection,” in Proceedings of the 2017 TRON Symposium, 2017.

[34] O. Chowdhury, L. Jia, D. Garg, and A. Datta, “Temporal Mode-Checking for Runtime Monitoring of Privacy Policies,” in Proceedings of the International Conference on Computer Aided Verification, 2014.

[35] T. Y. Chen et al., “Metamorphic Testing for Cybersecurity,” Computer (Long. Beach. Calif)., pp. 48–55, Jun. 2016.

[36] Y. Chen, C. M. Poskitt, and J. Sun, “Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System,” in Proceedings of the 2018 IEEE Symposium on Security and Privacy, 2018.

[37] S. T. Hamman, “Improving the Cybersecurity of Cyber-Physical Systems Through Behavioral Game Theory and Model Checking in Practice and in Education,” Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2016.

[38] D. Andriesse, H. Bos, and A. Slowinska, “Parallax: Implicit Code Integrity Verification Using Return-Oriented Programming,” in Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015.

[39] S. A. Asadollah, D. Sundmark, S. Eldh, and H. Hansson, “A Runtime Verification Tool for Detecting Concurrency Bugs in FreeRTOS Embedded Software,” in Proceedings - 17th International Symposium on Parallel and Distributed Computing, ISPDC 2018, 2018, pp. 172–179.

[40] W. Lu, S. Shu, H. Shi, R. Li, and W. Dong, “Synthesizing Secure Reactive Controller for Unmanned Aerial System,” in Proceedings of the 6th International Conference on Dependable Systems and Their Applications, 2019.

[41] L. Garcia, S. Zonouz, D. Wei, and L. P. de Aguiar, “Detecting PLC Control Corruption via On-Device Runtime Verification,” in Proceedings of the 2016 Resilience Week, 2016.

[42] J. Straub, “Mutual assured destruction in information, influence and cyber warfare: Comparing, contrasting and combining relevant scenarios,” Technol. Soc., vol. 59, p. 101177, Nov. 2019.