Demystifying Cyber-Physical Malware[*]

Suraj C. Kothari
Iowa State University and EnSoft Corp., Ames, Iowa 50011
kothari@iastate.edu

ABSTRACT

The traditional notion of malware is too narrow, and the prevalent characterizations (virus, worm, trojan horse, spyware etc.) are neither precise nor comprehensive enough to characterize cyber-physical malware (CPM). Detecting sophisticated CPM is like searching for a needle in the haystack without knowing what the needle looks like. This technical briefing congregates interdisciplinary knowledge to describe the fundamentals of CPM, the mathematical foundation for analyzing and verifying CPM, the current state-of-the-art, the challenges, and directions for future research. Employing real-world examples, we shall illustrate the challenges of analyzing and verifying CPM.

CCS CONCEPTS

• **Security and privacy → Software and application security**; • **Software and its engineering → Software verification and validation**;

KEYWORDS

Cyber-Physical Malware, Software Modeling and Verification

1 INTRODUCTION TO THE TOPIC

The imminent danger of cyber-physical malware (CPM) is evident from attacks such as the power outage in Ukraine [4] or the hijacking of a Jeep Cherokee [2]. The net-centricity of modern systems offers an adversary affordable attack vectors through cyberspace against critical missions. We are arguably at risk to an asymmetric attack vector launched by a terrorist adversary that cannot, or chooses not to confront in a conventional conflict. The Internet of Things (IoT) implies more software-driven devices and thus increased CPM risk.

Reliability is not security [13]. Imagine a GPS device that works accurately, except it malfunctions in Afghanistan on a full moon day. No matter how exhaustive the reliability testing in the factory, the GPS manufacturer will not catch this mission-critical flaw. Unlike the widely studied malware in the wild, sophisticated CPM remains in a stealth mode until a trigger activates it.

Software theorists are after automated model checking and theorem proving in order to verify software. Software practitioners are after perfecting the art of penetration testing. How can the theorists or the practitioners effectively apply their knowledge to CPM? If we were to use today's software verification tools based on *Formal Methods* (FMs) to verify the GPS software, how do we decide what to prove?

Prevalent research mainly focuses on monitoring and protecting the interfaces to the cyber-physical systems (CPS) with techniques such as intrusion detection. However, CPS security problems are often rooted in the complex CPS software. It is hard for the CPS community to understand intricacies of software analysis and verification. And for the software engineering community, the lack of adequate CPS knowledge is a major roadblock. This makes it important to demystify CPM, so that software engineers can model the CPM problems, establish the mathematical foundation, and advance the software analysis and verification techniques to effectively address the CPM problems.

Technological advances in computing, communications, and control, have set the stage for a next generation of CPS for energy, environment, transportation, and health care. The context for modeling CPM [7, 14, 15], need to be exposed so that the software engineering community can engage in collaborative interdisciplinary research for evolving CPM characterizations that can cover the vast expanse from mobile phones apps to Supervisory Control and Data Acquisition (SCADA) systems.

The technical briefing leverages our team's work on: (a) analyzing complex CPM on DARPA Automated Program Analysis for Cybersecurity (APAC) [1] and Space/Time Analysis for Cybersecurity (STAC) [3] programs, (b) developing commercial products to model and analyze control systems software for automobile, avionics, and other industries for whom safety and security is a major concern.

2 RELEVANCE TO SOFTWARE ENGINEERING

Software engineering community has an important role to play in developing comprehensive characterizations and use them to develop effective proactive measures to defend against CPM. Reactive measures such as issuing security patches or antivirus protection are unacceptable solutions, given the serious consequences of CPM. The knowledge about CPM gained through this technical briefing will be useful to understand the need for new modeling, analysis, and verification techniques. The real-world CPM examples from the technical briefing will bring out limitations of the current software security techniques and the need for new research directions. The experiment-discover paradigm mingled through the briefing will be of interest to innovate education in ways that arouse student's curiosity and creativity.

[*]This material is based on research sponsored by DARPA under agreement numbers FA8750-15-2-0080 and FA8750-12-2-0126. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3183459

3 TECHNICAL BRIEFING TOPIC

The briefing will be shaped from the perspective of *crucial needs* for modeling, analyzing, and verifying CPM.

Modeling: The CIA triad [16] characterizes the impact of the malware but it is not meant to facilitate analysis or verification of software. Modeling research is needed to characterize the program artifacts that enact CPM.

Modeling research is important in multiple ways. We shall focus on two crucial needs that modeling must serve. First, without a powerful abstraction, we are left to deal with an endless variety of problems. For example, without the abstraction of variables and the linear system of equations, there is an endless variety of constraint-satisfaction problems. Similarly, without effective modeling of CPM, we have an endless variety of physical systems as well as their varied malfunctions. Second, a powerful abstraction is necessary to avoid ad-hoc and inefficient solutions. For example, the abstraction of linear system of equations has enabled the powerful Gauss or Jacobi methods that scale to extremely large constraint satisfaction problems encountered in science and engineering. Similarly, modeling is a necessity to design efficient and accurate algorithms to analyze and verify CPM.

Analysis: Complete automation and machine learning are emphasized in many current research approaches to analyze software for security. The technical briefing will illustrate the shortcomings of such techniques to address CPM.

The CPM requires hypothesizing the potential malfunction and its trigger. Going back to the GPS example, the *fault* is the code that corrupts the location information, and the *trigger* is the conjunctive condition "Afghanistan and full moon." The trigger is so obscure that arriving at it by an automated analysis or machine learning is not likely. Moreover, CPM attackers design malware to make automated program analysis inordinately difficult. For example, the malware could be hidden in asynchronous processing so that automated data or control flow analyses succumb to inaccuracies that require inordinate human effort to weed out.

We will discuss the need for a human-on-the-loop approach [8] to address the challenges of hypothesizing and analyzing CPM. We will present the challenges for developing analysis techniques for an effective human-on-the-loop approach. We will present them as the challenges of *amplifying intelligence*, following up on the idea presented by Frederick Brooks in his Allen Newell address on solving complex problems [6].

As examples, we shall include lessons we have learnt and how they have shaped our research on DARPA projects to develop an effective human-on-the-loop approach for analyzing CPM. Specifically, we will discuss an approach that treats software as a graph database and deploys a query language to enable humans to gather evidence and drive CPM analysis interactively or programmatically [10, 12].

We shall illustrate the need to complement static with dynamic analysis and vice versa. Specifically, we will discuss *dynamically informed static analysis* and *statically informed dynamic analysis* as ways to combine both the analyses as a pragmatic approach to achieve scalability and accuracy.

Verification: Following up on the modeling, we will discuss software verification as a technique to prove properties formulated as a part of the model. We shall illustrate shortcomings of a completely automated approach to verification. Given the complexity of CPM and the possibility of catastrophic consequences, it is prudent to make the verification transparent so that a human can easily *participate* in verification, either to crosscheck the result or to complete the verification where automation falls short.

Following up on the papers [5, 9], we will differentiate between *machine-centric verification* and *human-centric verification* as two paradigms. We will relate the difference between the two paradigms to the transparency of verification. We will expound on the need for *encapsulation* and *modularity* to make automated proofs transparent by embedding high-level concepts in proofs and by bringing modularity to proofs.

Experiment-Discover Loop: The briefing will be mingled with the *experiment-discover loop* as an important theme to advance CPM research. Systematic experimentation is crucial for revolutionary research in science and engineering. It is hard to imagine that Newton could have thought of the gravitational force and the laws of motion without Galileo's experiments. Mathematicians perform experiments by working out examples [11]. We shall illustrate benefits of the experiment-discover loop for evolving powerful concepts necessary for efficient analysis and verification.

ACKNOWLEDGEMENT

We thank our colleagues from Iowa State University and EnSoft for their help with this paper. Dr. Kothari is the founder President and a financial stakeholder in EnSoft.

REFERENCES

[1] 2012. Automated Program Analysis for Cybersecurity (APAC). https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-11-63/listing.html. (2012).

[2] 2015. Hijacking a Jeep Cherokee. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. (2015).

[3] 2015. Space/Time Analysis for Cybersecurity (STAC). https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-14-60/listing.html. (2015).

[4] 2016. Ukraine Power Outage. http://reut.rs/1SYBTPq. (2016).

[5] Dirk Beyer, Matthias Dangl, Daniel Dietsch, and Matthias Heizmann. 2016. Correctness witnesses: Exchanging verification results between verifiers. In *International Symposium on Foundations of Software Engineering*. ACM, 326–337.

[6] Frederick P Brooks Jr. 1996. The computer scientist as toolsmith II. *Commun. ACM* 39, 3 (1996), 61–68.

[7] Eric Byres, P Eng, and Justin Lowe. 2004. The Myths and Facts behind Cyber Security Risks for Industrial Control Systems. In *Proceedings of the VDE Kongress*. 213–218.

[8] Mary Cummings. 2008. Supervising automation: humans on the loop. http://web.mit.edu/aeroastro/news/magazine/aeroastro5/cummings.html. (2008). Online; accessed 10-May-2017.

[9] Richard A De Millo, Richard J Lipton, and Alan J Perlis. 1979. Social processes and proofs of theorems and programs. *Commun. ACM* 22, 5 (1979), 271–280.

[10] Tom Deering, Suresh Kothari, Jeremias Sauceda, and Jon Mathews. 2014. Atlas: a new way to explore software, build analysis tools. *International Conference on Software Engineering* (2014).

[11] David Epstein and Sylvio Levy. 1995. Experimentation and proof in mathematics. *Notices of the AMS* 42, 6 (1995), 670–674.

[12] Benjamin Holland, Tom Deering, Suresh Kothari, Jon Mathews, and Nikhil Ranade. 2015. Security Toolbox for Detecting Novel and Sophisticated Android Malware. *International Conference on Software Engineering* (2015).

[13] Kamal Jabbour and Sarah Muccio. 2011. The science of mission assurance. *Journal of Strategic Security* 4, 2 (2011), 61.

[14] Arash Nourian and Stuart Madnick. 2015. A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet. *IEEE Transactions on Dependable and Secure Computing* (2015).

[15] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. 2013. Attack detection and identification in cyber-physical systems. *IEEE Trans. Automat. Control* 58, 11 (2013), 2715–2729.

[16] Chad Perrin. 2008. The CIA triad. *http://www.techrepublic.com/blog/security/the-cia-triad/488* (2008).