# A configurable approach to cyber-physical systems fuzzing

Guillaume Nguyen
guillaume.nguyen@unamur.be
Fac. of Computer Science - NADI, University of Namur
Namur, Belgium

## ABSTRACT

Operational Technology has gotten a growing place in our daily lives. With the increasing number of devices (connected or not), the need for a clean environment that allows effective and efficient testing is also increasing. Furthermore, some devices are connected to the physical world with the ability to affect it. Assembling those specific devices with at least a sensor, an actuator, and a (micro)processor creates Cyber-Physical Systems (CPSs). With such power in the hands of machines, it is imperative that they behave as expected and that they resist disruptive environments (whether from cyber attacks, unwanted noise, or environmental disturbance). Indeed, the impacts of an unexpected behavior could lead to significant damage (disruption of the production line, overheating of a nuclear reactor, false fire alarm, etc.). That is why the safety and the security of those systems should also be at the center of concerns. As the definition of those systems is quite simple, one can assemble various components to create a unique CPS. One could also modify an existing CPS to satisfy a specific need (e.g., a fire alarm system modified to detect carbon monoxide in the air, changing communication protocols or programming languages used for the sake of maintainability). To test such highly-configurable systems, there are multiple techniques. Fuzzing works particularly well with any system by sending pseudo-random inputs. To adapt to specific systems and test requirements (coverage, resources, etc.), fuzzing is itself highly-configurable (Grammar-based, symbolic, probabilistic, etc.). This is why it could perform particularly well with CPSs, which all might require a different and specific testing approach depending on their interfaces, components, etc. Currently, no frameworks allow for the classification of CPSs to enable the automatization of the generation of tests following their requirements. That is why this thesis will take a configurable approach to find and recommend the most suitable classification of CPS for testing and comparing the various fuzzing techniques to find the most effective ones based on relevant features and requirements of CPSs.

## CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; • **Software and its engineering → Software testing and debugging**.

## KEYWORDS

CPS, testing and quality assurance, fuzzing, embedded systems, IoT

**ACM Reference Format:**
Guillaume Nguyen. 2023. A configurable approach to cyber-physical systems fuzzing. In *27th ACM International Systems and Software Product Line Conference - Volume B (SPLC '23), August 28-September 1, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3579028.3609005

## 1 INTRODUCTION AND MOTIVATION

For years now, information and communication technology (ICT) has taken a growing place in our physical daily lives. Indeed, even more objects are now connected to the Internet of Things (IoT) [14] and this number is expected to keep growing [2]. Furthermore, objects such as cars [3] and (smart) factories [18] that were previously not connected are now being introduced. Those interconnected entities also have the ability to interact with the real world. Indeed, they compute information from the real world (virtually) which could result in an action in the said world (e.g., a connected pacemaker detecting a heart failure and sending an electrical discharge). This specific kind of entity is called a Cyber-Physical System (CPS) [19]. Similar to the software product line engineering phylosophy, those systems are built by assembling various reusable components [6, 17, 37]: for instance, a temperature sensor could be used to monitor a connected greenhouse in order to maintain a specific temperature with the ability to activate a cooling or heating system, while it could also be used to monitor and manage the tempering of chocolate in a biscuit factory.

Here are a few examples of those systems in various domains of application. (1) In healthcare, wireless medical devices presence is growing in hospitals and operating rooms [22]. (2) Smart factories through Industry 4.0 aim at increasing the efficiency of the product line either by reducing the costs or improving the flexibility of the resources by using interconnected devices, sensors, and actuators [15, 18]. (3) In aerospace, Ingenuity Mars copter had to adapt itself to the aerodynamics of Mars to perform the first successful flight there [34]. (4) In the automotive industry, advanced driver assistance systems (ADAS), alongside other technologies, aim at self-driving and connected cars [3].

As we can see, CPSs are omnipresent in our world and can be highly critical, as vulnerabilities leading to failure could imply significant damages in the real (physical) world [30]. Nevertheless, this type of system might have been previously discussed in the past as *Embedded Systems*. An embedded system is used to implement dedicated functions, is comprised of microprocessors, and might use sensors and/or actuators. Usually, no or few changes are allowed after installation. Due to their design, the correctness of those systems is highly important as they are mainly used to perform safety-critical tasks [10]. Based on that definition, one could consider that embedded systems are components of a CPS. Circling back to the previously introduced interconnection of CPSs (alongside connections to the internet), we can say that it increases the potential number of vulnerabilities among components. Thus, testing and quality assurance of those CPSs in design and production is of uttermost importance. However, creating holistic tests (for each component and usage) on short notice is a challenge in itself. Using randomly generated input is an approach that could leverage this level of coverage within an allocated limit of time and resources while not being specific (and widely usable). This type of testing is called *Fuzzing* [25] and is one of the most widely used techniques to find vulnerabilities.

Fuzzing has been widely used in many different ways before and since its first appearance in the literature by Barton Miller [25] in the early 1990s. The origin of the word started during a stormy night while the storm was causing noise on the line. Indeed, this noise added random characters to the command when it was not sent quickly enough. The interesting observation was that sometimes the program crashed on the other side. In the days following this experience, B. Miller asked his students to look into this type of testing on the UNIX system [32]. It has been found to enable a tester or a developer to detect input-based problems that would not have been thought of during development.

Since then, this specific type of testing tool has evolved into smarter versions [40] with features such as the use and mining of grammars [39], code coverage techniques [12], parsing techniques [36], probability rules [35], and a combination of all these techniques [8] to test systems more efficiently. Indeed, simply generating random characters in a more or less long string and sending it as input to specific or nonspecific programs uses an enormous amount of resources to find potential errors in the code. However, when eliminating known sources of rejection (e.g., when communicating with specific protocols by automatically adding corresponding headers), one can send *fuzz* (i.e., a random input) in a shape that will always be interpreted by the targeted program thus, allowing more efficient usage of the resources. Of course, this is only an example of what can be done already to perform fuzzing, but it opens a vast world of optimization for this originally *brute-force* type of testing [31].

Nowadays, there are multiple fuzzing tools or *fuzzers* that are being developed, tested, and compared against a set of test programs [5]. However, these are almost all used in nonspecific ways. For example, AFL is the go-tool when thinking about fuzzing without regards to the type of system (software, hardware, network) tested, the targeted system just need to have an interface for AFL to start fuzzing. Of course, an image fuzzer for adversarial training of machine learning models [27] will not be the same as Command-Line Interface (CLI) fuzzers. Still, except for particular tools, there are no holistic guidelines on how and when (and why) to use specific fuzzing techniques.

The purpose of this thesis is to rely on a highly-configurable approach to recommend the use of the most efficient fuzzing techniques and tools based on the identified CPS alongside a set of testing requirements (time, knowledge, and resources).

## 2 BACKGROUND

### 2.1 Cyber-Physical Systems

This section is divided in three to give an insight on how we intend to tackle our problem in a timely manner. First we need to define and classify CPS in order to give a proper scope of our work. Then, we need to find a way to identify a system and properly classify it before suggesting tests. The main goal is to see if we are able to identify any system base on our classification scheme. Finally, we will compare existing fuzzing tools to determine which ones would be the bests for each CPS or CPS component.

*2.1.1 Definition.* The concept of Cyber-Physical System (CPS) shows first traces of appearance in the literature in 2006 during a workshop carried out by the National Science Foundation (NSF) of the United States. This workshop led to the creation of three main founding papers from Lee [19], Sha et al. [29] and Rajkumar et al. [28]. The definition that we hold on to would be the following: "*Cyber-physical systems (CPS) are physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core. This intimate coupling between the cyber and physical will be manifested from the nano-world to large-scale wide-area systems of systems. The internet transformed how humans interact and communicate with one another, revolutionized how and where information is accessed, and even changed how people buy and sell products. Similarly, CPS will transform how humans interact with and control the physical world around us.*" [28, p. 1]

Furthermore, IoT is a "*concept used to define or reference systems that rely on autonomous communication of a group of physical objects*" [14]. We can see that the concepts of IoT and CPS (and embedded systems) are relatively close, which is why all should be considered when trying to define and classify CPS. One could consider that embedded systems are components of CPS that make use of IoT to communicate. When bringing CPS and IoT together we have the following definition: *CPS is a controllable, credible, and scalable network physical equipment system which deeply integrates the ability of computing, communication and control on the basis of information acquisition in IOT. Through the feedback loop of the interaction between calculation process and physical process, deep integration and real-time interaction is realized to increase or to extend new function, so that a physical entity can be detected or controlled in a safe, reliable, and efficient way.* [21, p. 28] We rely on this widely accepted definition of CPS before performing further research. As we can see, researchers tend to agree on the definition of CPS so we can go further into identifying the characteristics of a CPS.

*2.1.2 Classification.* The University of Berkeley provides us with a concept map around CPS. This map describes a CPS as being feedback system that requires cyber-security, safety, improved design tools and design methodology, and has an application in a specific domain [1]. Pushing further the description of a CPS, Tekinerdogan et al. [33] provide us with a metamodel (Figure 1) and a feature-based ontology. They describe a CPS as having constituent elements, non-functional requirements, an application domain, a discipline (software engineering, civil engineering, etc.), and an architecture. The constituent elements of a CPS are the following [33]:

- **Cyber** - element controlling and communicating with the other elements
- **Control** - state, disturbance, input, output, goal, feedback, dynamics, properties, diagnostics, prognostics
- **Human** (non-mandatory) - role, event, entity, action
- **Network** - configuration and communication

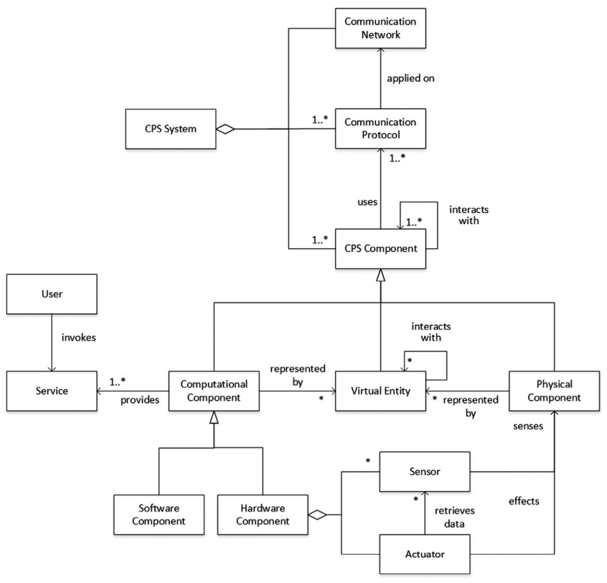

**Figure 1: The metamodel by Tekinerdogan et al. [33, p. 47]**

- **Physical** - sensors, actuators, plant, controller and environment

With all this information we can now continue towards the classification of those systems. For now, the primary classification of CPSs we found is based on the industry in which they are used. Cardin provided us with a framework to classify Cyber-Physical Production Systems (CPPSs) which are the CPSs specific to production lines [11]. Miller et al. tried to establish classification rules for cybersecurity incidents around CPSs [24]. However, there is still a need for a more general classification of CPSs, which should allow for easier test generation and orchestration. To tackle this problem we will analyse CPS from as many domains of applications as we can and try to determine where similarities and differences lie when talking about testing.

## 2.2 System identification

Suggesting tests for a specific CPS requires considering various situations. Indeed, we could face the need to identify a CPS without any access to the blueprint (source code, hardware used, etc.) also referred to as *Black Box* where we only have access to the system as it is meant to be used. Access to the entire blueprints would definitely reduce the time needed to identify a CPS; this is called *White Box*. However, we could also have full access to the physical device and use identification tools or penetration testing (*pen-test*) tools to approximate (*mine*) the design choices that were made, this is referred to as *Grey Box*. In any case, the identification of the CPS is a necessary step. Extrapolating from the IoT for the CPS connected to a network, we can see that researchers have found ways to identify such devices (known or unknown). Ortiz et al. trained a model to recognize specific packets on the network. Their model was able to identify pairs of IoT devices communicating with each other and they found that those devices had quite a similar signature which was different from non-IoT (NoT) devices [26]. However, their model was less accurate when trying to identify devices not previously seen as some IoT and NoT devices showed a statistically similar signature. Afterwards, Bremler-Barr et al. worked on a new model based on three classifiers (Traffic, DHCP, and a unified one using both previous classifiers). After selecting network traffic and DHCP features to train their models, they were able to achieve a higher accuracy than other researchers in correctly classifying devices as IoT or NoT. The unified classifier was the best performer by using the traffic features classifier and DHCP features when available to improve accuracy [9].

Those are only examples of attempts to identify IoT and NoT devices on the network that could be used to do the same with a CPS. However, CPSs might not be connected to the internet. Furthermore, even interconnected CPSs could be using proprietary protocols that would not be understood or detected by previously introduced methods. [4]

Focusing more on testing for CPS, Babun et al. [4] reported three key features for usually identifying a CPS: the type of task performed, the resources available to the CPS and the timing properties and precision of the message processing and emission. To identify the class of devices, they also identified three features: device metrics, device behavior, and device performance. They analyzed the possibilities to spoof (impersonate) a device in order to disrupt a critical infrastructure assuming they were communicating on a network. They proposed *Stop & Frisk* (S&F) which is a "*signature-based fingerprinting approach that performs CPS device-class identification*" [4]. This identification method was more accurate than previous work on the matter. To help us solve this problem we will first look into which resources (codes, proprietary protocols and so on) are available and where. Indeed, we believe that manufacturers are sometimes not willing to share all information about a component or a system to users or even clients. In a second time, based on those findings we will explore the available methods to gain more information about a system on one hand and look at the fuzzing methods available when working with not fully identified systems on the other hand.

## 2.3 Comparison and classification of fuzzers

Researchers have already tried to come up with a classification and description for fuzzers based on the technique used and the information received from the tested system. They aimed to provide state-of-the-art vulnerability discovery mechanisms in place together with a more in-depth focus on fuzzing techniques and subsequent [20]. An algorithm has also been developed to automatically classify fuzzers with 16 characteristics [23]. This research provides a chronological view of the various fuzzers that have been developed since the first paper about fuzzing in the 1990s. Subsequently, Boehme et al. [7] summarized the remaining challenges about fuzzing for the coming years and confronted those challenges in a survey spread amongst other researchers. They established that many challenges are still awaiting to be tackled to make fuzzing more efficient. They separated the challenges into the following categories: technique automation, human component, fuzzing theory, evaluation, and benchmark.

Concerning the benchmark, Beaman et al. compared various fuzzers in categories based on the technique used (Markov chain, binary instrumentation, etc.) [5] which gives an interesting overview over the current capabilities of existing fuzzers. More interestingly, Eceiza et al. give a summary of the classification of embedded systems and of fuzzing techniques while challenging the current choices of fuzzers based on the operating system (OS). They found that there was several flaws in the conception of fuzzers concerning the testing of embedded systems (IoT and CPS) [13]. Furthermore, there are many books explaining the various fuzzing techniques [40], the tests [32], and fuzzing in general [31].

More recently, Yun et al. provide us with a view of fuzzing adapted to the needs of embedded systems. They reviewed the current literature in order to find the most suitable fuzzing techniques and reveal future challenges for the fuzzing of this type of systems [38].

As we can see, the main approach for the classification and comparison is mostly oriented towards the techniques used amongst the same types of fuzzer and less oriented towards the tested system and test requirements (time and resources constraints, design testing vs production testing, etc.). Tackling this problem is the last goal of this thesis. Based on the classification and the identification of CPS we will compare the performance of the various available fuzzers within a set of constraints.

## 3 RESEARCH QUESTIONS

The thesis will answer the following research questions:

**RQ1. How can we classify CPS to perform efficient testing?** Based on the CPS definition, creation of a classification scheme oriented toward the orchestration of tests which current classifications fail to provide. This will be used to provide a robust test framework.

**RQ2. What are the best fuzzing techniques for each CPS type?** Compare fuzzers and fuzzing techniques following the CPS classification to produce a recommendation framework. Based on the many fuzzing techniques, we'll identify the most relevant ones. The main metrics will come from research papers suggesting a fuzzing tool and comparing it to other ones using a test set.

**RQ3. What is missing from the fuzzer landscapes to have a holistic fuzzing toolbox for CPS?** Identify the research opportunities from the production of the framework.

**RQ4. Which type of non-existing fuzzer would contribute the most to the testing of CPS?** Start developing a missing fuzzer.

In a nutshell, this thesis will have four main steps. The first step will be focused on understanding and laying down the context with the classification of CPSs (**RQ1**). Then, we will establish the list of possible automated testing approaches for the different CPSs with a focus on fuzzing and comparing the available tools (fuzzers) (**RQ2**). Afterwards, we will look into what the missing fuzzers are to be able to perform tests on all the different CPSs in the classification framework (**RQ3**). Finally, we will develop a fuzzer missing from the current landscape that will contribute to the tools available for testing CPSs (**RQ4**).

## 4 RESEARCH METHODOLOGY AND APPROACH

We will use a *Design Science* approach with case studies carried on at industrial project partners' site. This research has a strong industrial orientation with a primary focus on Belgium. In order to provide testers with a useful test framework for CPS we will carry out *interviews* and a large *survey* that we will send out to various industrial actors which are actual users of CPSs. This will allow us to perceive the most realistic view of the challenges related to testing CPSs in those various industries. Afterwards, we will try to propose a classification framework for CPS that will ease the testing of those systems within specific contexts. To test this framework we will classify CPSs from different industries and see how they fit. Then, for each type of CPS (or CPS component depending on the classification) in the framework we will try to find the best working fuzzing techniques by comparing the capable fuzzers among each other using meaningful criteria such as the vulnerabilities found or the time and resources needed for the test. Finally, we will establish the list of missing fuzzers for testing the whole classification framework based on the previous analysis of the literature and available tools we realised when comparing the fuzzers among each other. As an extended contribution we will work on reducing the list of missing fuzzers from our list by creating a relevant one. The framework produced during the thesis will be empirically validated together with the industrial partners of the project and on existing benchmarks [16].

## 5 PRELIMINARY RESULTS

The research started in September 2022. We currently do not have concrete results yet, but we shared a survey to various industrial actors and the results should arrive in the coming months. The survey was mainly focused at the discovery on how the users of CPS were dealing with testing their systems. It was build following open interviews with an industrial actor and four researchers from two Belgian research centers working on CPS. Is there any testing? Where does this testing happens? Do the manufacturers provide test results or testing tools? Are examples of questions we try to answer. Given the industrial emphasis of this study, we plan to submit the results to an industry track before the end of the summer. Following the project plan this research will have a strong industrial relationship. Although it will not be an *Action Research*, it will share most of its markers by pushing the research following industrial feedback.

## 6 WORK PLAN

During the first year I will Continue to review the literature on CPS, embedded systems, IoT, real time systems, testing and quality assurance and send out a survey to industrial actors. I will also try to publish the classification scheme based on the survey. During the second year, I will try to find the most fitting testing methods based on the classification scheme and compare fuzzers among each other using studies carried out by other researchers to determine the most fitting ones. I will attempt to publish the results. During the third year I will try to figure if there are missing fuzzers that would benefits to CPSs on the classification scheme and prototype an automated testing tools for CPS (based on its identification and classification on the scheme). I will try to publish the list of missing fuzzers and the automated tool. During the fourth year I will start to compile my thesis and create a fuzzer myself that I will try to publish.

## ACKNOWLEDGMENTS

This research was funded by the CyberExcellence by DigitalWallonia project (No. 2110186), funded by the Public Service of Wallonia (SPW Recherche).

## REFERENCES

[1] [n. d.]. *Cyber-physical systems, a Concept Map.* https://ptolemy.berkeley.edu/projects/cps//

[2] Tanweer Alam. 2018. A Reliable Communication Framework and Its Use in Internet of Things (IoT). 3 (05 2018).

[3] Fabio Arena, Giovanni Pau, and Alessandro Severino. 2020. An overview on the current status and future perspectives of smart cars. *Infrastructures* 5, 7 (June 2020), 53.

[4] Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2021. CPS Device-Class Identification via Behavioral Fingerprinting: From Theory to Practice. *IEEE Transactions on Information Forensics and Security* 16 (2021), 2413–2428. https://doi.org/10.1109/tifs.2021.3054968

[5] Craig Beaman, Michael Redbourne, J. Darren Mummery, and Saqib Hakak. 2022. Fuzzing vulnerability discovery techniques: Survey, challenges and future directions. *Computers & Security* 120 (Sept. 2022), 102813. https://doi.org/10.1016/j.cose.2022.102813

[6] Maurice H. ter Beek, Alessandro Fantechi, and Stefania Gnesi. 2018. Product line models of large cyber-physical systems: the case of ERTMS/ETCS. In *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1.* ACM, Gothenburg Sweden, 208–214. https://doi.org/10.1145/3233027.3233046

[7] Marcel Boehme, Cristian Cadar, and Abhik Roychoudhury. 2021. Fuzzing: Challenges and Reflections. *IEEE Softw.* 38, 3 (2021), 79–86.

[8] Marcel Bohme, Van-Thuan Pham, and Abhik Roychoudhury. 2019. Coverage-Based Greybox Fuzzing as Markov Chain. *IEEE Transactions on Software Engineering* 45, 5 (May 2019), 489–506. https://doi.org/10.1109/tse.2017.2785841

[9] Anat Bremler-Barr, Haim Levy, and Zohar Yakhini. 2020. IoT or NoT: Identifying IoT Devices in a Short Time Scale. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium.* IEEE. https://doi.org/10.1109/noms47738.2020.9110451

[10] Raul Camposano and Jorg Wilberg. 1996. Embedded system design. *Des. Autom. Embed. Syst.* 1, 1-2 (Jan. 1996), 5–50.

[11] Olivier Cardin. 2019. Classification of cyber-physical production systems applications: Proposition of an analysis framework. *Computers in Industry* 104 (Jan. 2019), 11–21. https://doi.org/10.1016/j.compind.2018.10.002

[12] Peng Chen and Hao Chen. 2018. Angora: Efficient Fuzzing by Principled Search. In *2018 IEEE Symposium on Security and Privacy (SP).* IEEE. https://doi.org/10.1109/sp.2018.00046

[13] Maialen Eceiza, Jose Luis Flores, and Mikel Iturbe. 2021. Fuzzing the Internet of Things: A Review on the Techniques and Challenges for Efficient Vulnerability Discovery in Embedded Systems. *IEEE Internet of Things Journal* 8, 13 (July 2021), 10390–10411. https://doi.org/10.1109/jiot.2021.3056179

[14] Jordán Pascual Espada, Ronald R. Yager, and Bin Guo. 2014. Internet of things: Smart things network and communication. *Journal of Network and Computer Applications* 42 (June 2014), 118–119. https://doi.org/10.1016/j.jnca.2014.03.003

[15] Aitziber Iglesias, Hong Lu, Cristóbal Arellano, Tao Yue, Shaukat Ali, and Goiuria Sagardui. 2017. Product Line Engineering of Monitoring Functionality in Industrial Cyber-Physical Systems: A Domain Analysis. In *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A.* ACM, Sevilla Spain, 195–204. https://doi.org/10.1145/3106195.3106223

[16] Sangeeth Kochanthara, Yanja Dajsuren, Loek Cleophas, and Mark van den Brand. 2022. Painting the landscape of automotive software in GitHub. In *Proceedings of the 19th International Conference on Mining Software Repositories.* ACM, Pittsburgh Pennsylvania, 215–226. https://doi.org/10.1145/3524842.3528460

[17] Jacob Krüger, Sebastian Nielebock, Sebastian Krieter, Christian Diedrich, Thomas Leich, Gunter Saake, Sebastian Zug, and Frank Ortmeier. 2017. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A.* ACM, Sevilla Spain, 237–241. https://doi.org/10.1145/3106195.3106217

[18] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Bus. Inf. Syst. Eng.* 6, 4 (Aug. 2014), 239–242.

[19] Edward Lee. 2007. Computing Foundations and Practice for Cyber Physical Systems: A Preliminary Report. (01 2007).

[20] Jun Li, Bodong Zhao, and Chao Zhang. 2018. Fuzzing: a survey. *Cybersecurity* 1, 1 (June 2018). https://doi.org/10.1186/s42400-018-0002-y

[21] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao, and Zihe Liu. 2017. Review on cyber-physical systems. *IEEE/CAA J. Autom. Sin.* 4, 1 (Jan. 2017), 27–40.

[22] Mohamed R Mahfouz, Gary To, and Michael J Kuhn. 2012. Smart instruments: Wireless technology invades the operating room. In *2012 IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems (BioWireleSS)* (Santa Clara, CA, USA). IEEE.

[23] Valentin J.M. Manes, HyungSeok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele, Edward J. Schwartz, and Maverick Woo. 2021. The Art, Science, and Engineering of Fuzzing: A Survey. *IEEE Transactions on Software Engineering* 47, 11 (Nov. 2021), 2312–2331. https://doi.org/10.1109/tse.2019.2946563

[24] Bill Miller, Dale Rowe, Richard Helps, and Ross Woodside. 2014. A Comprehensive and Open Framework for Classifying Incidents Involving Cyber-Physical Systems.

[25] Barton P. Miller, Lars Fredriksen, and Bryan So. 1990. An empirical study of the reliability of UNIX utilities. *Commun. ACM* 33, 12 (Dec. 1990), 32–44. https://doi.org/10.1145/96267.96279

[26] Jorge Ortiz, Catherine Crawford, and Franck Le. 2019. DeviceMien. In *Proceedings of the International Conference on Internet of Things Design and Implementation.* ACM. https://doi.org/10.1145/3302505.3310073

[27] Yi Qin and Chuan Yue. 2022. Fuzzing-based hard-label black-box attacks against machine learning models. *Computers & Security* 117 (June 2022), 102694. https://doi.org/10.1016/j.cose.2022.102694

[28] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. 2010. Cyber-physical systems. In *Proceedings of the 47th Design Automation Conference* (Anaheim California). ACM, New York, NY, USA.

[29] Lui Sha, Sathish Gopalakrishnan, Xue Liu, and Qixin Wang. 2008. Cyber-Physical Systems: A New Frontier. In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)* (Taichung, Taiwan). IEEE.

[30] Abdelkader Magdy Shaaban, Thomas Gruber, and Christoph Schmittner. 2019. Ontology-Based Security Tool for Critical Cyber-Physical Systems. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B.* ACM, Paris France, 207–210. https://doi.org/10.1145/3307630.3342397

[31] Michael Sutton, Adam Greene, and Pedram Amini. 2007. *Fuzzing.* Addison-Wesley Educational, Boston, MA.

[32] Air Takanen, Jared D. Demott, Charles Miller, and Atte Kettunen. 2018. *Fuzzing for Software Security Testing and Quality Assurance, Second Edition.*

[33] Bedir Tekinerdogan, Dominique Blouin, Hans Vangheluwe, Miguel Goulão, Paulo Carreira, and Vasco Amaral. 2020. *Multi-Paradigm Modelling approaches for cyber-Physical Systems.* Academic Press, San Diego, CA.

[34] Theodore Tzanetos, Mimi Aung, J Balaram, Havard Fjrer Grip, Jaakko T Karras, Timothy K Canham, Gerik Kubiak, Joshua Anderson, Gene Merewether, Michael Starch, Mike Pauken, Stefano Cappucci, Matthew Chase, Matthew Golombek, Olivier Toupet, Marshall C Smart, Stephen Dawson, Erick Blandon Ramirez, Johnny Lam, Ryan Stern, Nacer Chahat, Joshua Ravich, Robert Hogg, Benjamin Pipenberg, Matthew Keennon, and Kenneth H Williford. 2022. Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout. In *2022 IEEE Aerospace Conference (AERO)* (Big Sky, MT, USA). IEEE.

[35] Junjie Wang, Bihuan Chen, Lei Wei, and Yang Liu. 2017. Skyfire: Data-Driven Seed Generation for Fuzzing. In *2017 IEEE Symposium on Security and Privacy (SP).* IEEE. https://doi.org/10.1109/sp.2017.23

[36] Dingning Yang, Yuqing Zhang, and Qixu Liu. 2012. BlendFuzz: A Model-Based Framework for Fuzz Testing Programs with Grammatical Inputs. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.* IEEE. https://doi.org/10.1109/trustcom.2012.99

[37] Tao Yue, Shaukat Ali, and Bran Selic. 2015. Cyber-physical system product line engineering: comprehensive domain analysis and experience report. In *Proceedings of the 19th International Conference on Software Product Line.* ACM, Nashville Tennessee, 338–347. https://doi.org/10.1145/2791060.2791067

[38] Joobeom Yun, Fayozbek Rustamov, Juhwan Kim, and Youngjoo Shin. 2023. Fuzzing of embedded systems: A survey. *ACM Comput. Surv.* 55, 7 (July 2023), 1–33.

[39] Michal Zalewski. 2014. *American Fuzzy Lop.* https://lcamtuf.coredump.cx/afl/

[40] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. 2022. The Fuzzing Book. In *The Fuzzing Book.* CISPA Helmholtz Center for Information Security. https://www.fuzzingbook.org/html/00_Table_of_Contents.html Retrieved 2022-05-18 13:13:27+02:00.

Received 2 May 2023; accepted 2 July 2023