Combining Binary Security, Artificial Neural Network Security and Philosophy of Science

Pedro Mendes da Silva [*]
pedro.mendes.da.silva@tecnico.ulisboa.pt

Abstract

Three papers from different disciplines are analyzed and a future research direction combining concepts from these papers is proposed, merging ideas from Binary Security, Neural Network Security and Philosophy of Science. [1]

1 Summary of paper proposals

Stephens et al. [1] proposed Driller, in the context of binary security. It is an Automated Binary Vulnerability Excavation (ABVE) system, which combines previous techniques, namely, a genetic input-mutating fuzzer and a selective concolic execution engine to identify deep bugs in binaries, while avoiding the limitations of each technique. It tries to avoid problems related to the fuzzer's need for input test cases and concolic execution engines typically succumbing to path explosion. It can detect any vulnerability that can lead to a program crash. Additionally, it supports arbitrary vulnerability specifications and it does not require any input test cases.

Carlini and Wagner [2] proposed a new way of evaluating the robustness of Neural Networks (NN) to adversarial inputs, in the context of NN vulnerability research. The authors developed new attacks and demonstrated that every existing defensively secured NN model was vulnerable to these new attacks. They proposed that to evaluate the robustness of a secured NN to adversarial inputs, designers should check for two properties. They state that designers should check whether the secured NN can resist a powerful attack. Furthermore, they argue that designers should construct adversarial inputs that break an unsecured model and should show that these inputs fail to break the secured model.

Herley and van Oorschot [3] proposed some ideas to advance security research in a more scientific way. They argue that many insights and methods from philosophy of science remain largely unexplored in security research. These include acknowledging the inductive claim and deductive claim differences and stopping relying on unfalsifiable claims, which are commonly used to justify many defensive measures, without of evidence of efficacy. Furthermore, they argue that security research should bring theory into contact with observation, by performing experiments in the real-world, using full-stack solutions, as opposed to making claims about isolated theoretical components, while disregarding their real-world implementations.

2 Critical examination

Driller [1] was proposed as the combination of two existing techniques, a genetic input-mutating fuzzer and a concolic execution engine, to avoid some limitations with these techniques. Fuzzers typically suffer from the need of defining input test cases. They can get stuck, while going through input mutations, without identifying new interesting execution paths. On the other hand, concolic execution engines make use of program interpretation and constraint solving techniques, for searching an executable's state space for vulnerabilities, which does not scale well to large programs, because it leads to a path explosion. Thus, the combination tries to avoid the concolic execution succumb to path explosion, while providing the fuzzer automatically with input cases.

Rawat et al. [4] argued that Driller's approach does not scale well to large executables, weakening one of fuzzing's original strengths, scalability. They proposed a different feedback loop, based on prioritizing deep execution paths, when mutating inputs, avoiding the use of symbolic execution. "Model-based whitebox fuzzing" [5] tries to improve on Driller by using information about the input file format to guide the input mutations. "Coverage-based Greybox Fuzzing" (CGF) [6] was proposed as an approach that requires no symbolic execution, as opposed to Driller, working by random mutation of inputs and selection of those that exercise interesting paths. Shoshitaishvili et al. [7] proposed an extendable framework for systematized binary analysis, implementing and combining several previous approaches, including Driller's.

Shoshitaishvili et al. [7] have pointed out that often the only feasible way to prove properties about the code that is actually executed is binary analysis. Thus, Driller could be used to check compliance to externally-imposed safety regulations, by suitably changing its vulnerability definitions. Similarly, it could be used to automatically check compliance with software quality properties, in the context of Software Engineering [8].

While NNs have become increasingly effective in several domains, such as image recognition, they are still vulnerable to adversarial attacks. A measure for characterizing how vulnerable a given NN is was needed. NN robustness is as measure of how easy it is to find adversarial inputs that are close to their original input, while producing a different output. Previous mea-

[*]INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, IST-Taguspark, 2744-016 Porto Salvo, Portugal

[1]Technical Report for the Interdisciplinary Research discipline, PDEIC, 2018.

sures [9] proved not to be effective, as Carlini and Wagner [2] developed new attacks which could break all existing defensive mechanisms, while suggesting that any NN robustness measure should include the results of a powerful attack, to evaluate the robustness of a secured NN model directly. Furthermore, they argue that a NN robustness measure should consider whether successful attacks on an unsecured NN model are successful in the secured one. Both defensive and offensive new security research followed. Feature Squeezing [10] was proposed for image classification, by reducing the color bit depth of each pixel and by spatial smoothing, as promising defensive mechanisms for NNs. Region-based classification was proposed [11] to achieve higher NN robustness than point-based classification. Results from Carlini and Wagner [2] were used to show that Google's Cloud Vision API is not robust to adversarial noise [12].

A NN robustness measure could be used to check compliance to safety properties imposed by regulations. Software Engineering could benefit from the inclusion of NN robustness testing and measures, as part of development processes [8].

A lack of scientific methodology in some security advices makes them poorly effective [3], as they frequently rely on unfalsifiable claims to justify many defensive measures, without evidence of their efficacy. Security community members frequently ignore the distinction between inductive and deductive claims, and improperly generalize theoretical security guarantees over isolated components to real-world integrated full-stack systems. A lack of training in experimental science or scientific methods by these members was also identified as a problem. NIST SP 800-63 [13] is commonly cited as the primary recommendation for complex password composition policies, and for password expiration policies. Science-oriented research has contributed to the 2017 revision [14] of this recommendation, which withdrew support for both password policies. By comparing the recommendations with empirical studies of real-world attacks, several researchers concluded that both password policies did not provide any real benefit [15, 16, 17].

Security community members would benefit from more knowledge about the philosophy of science. This would increase the probability of improving security research outcomes in the real world, by performing empirical studies, while opening the door for feedback and model correction. Science itself is a real-world endeavor, open to attacks [18]. Incorporating a full-stack of actors in the scientific endeavor models, while deriving new scientometric indicators from them, could be an interesting possibility. These could include probabilities of fraud, ghost-writing [19] and censorship. In yet another field, economics, an empirical study, from as early as 1956 [20], demonstrated that one of economic theory's basic assumptions, human rationality, did not hold empirically. Feedback and model correction was lost, as economics courses still assume human rationality to be true [21, 22, 23].

3 Future research direction

Running an ABVE tool like Driller over full-stack systems containing NN-based classifiers, could be a promising approach for automatically searching for vulnerabilities in such systems, possibly increasing confidence in their safety properties. This approach would combine ABVE [1], NN robustness evaluation [2] and full-stack testing [3].

The set of vulnerabilities supported by an ABVE tool could be extended. New vulnerabilities could be defined using a NN robustness measure, similar to the one proposed by Carlini and Wagner [2]. Such measure could be used to extend ABVE to the NN robustness domain. An ABVE tool would be augmented with a NN robustness measure (R_{NN}) calculation function and its respective lower threshold (LT). These could then be used to define a new known vulnerability condition, $R_{NN} < LT$ for a given input. The tool would then be able to search for such vulnerabilities in a given input binary running a real NN-classifier, by varying the inputs to this binary. These inputs could include, for example, two images, which would be mutated by the tool, until a vulnerability was found, as, for example, when the images are very similar but produce different NN-classification results.

Carlini and Wagner [2] define a NN-classifier analytically, as a mathematical function, and develop attacks by solving optimization problems over an isolated NN-classifier component. In a real-world scenario, however, the NN will usually be just one of the components of a larger system, running one or more real-world binary executables. Herley and van Oorschot [3] suggested that we should also evaluate these security properties in a real full-stack integrated system. For example, if we consider a self-driving vehicle [24], we should test the whole integrated system. Both hardware and software components should be included, such as front, rear and side cameras in the mirrors, other sensors like Radar and GPS, steering wheel controller, as well as the NN-based image classification software. However, it is difficult to use ABVE in a full-stack scenario, because most devices receive input from the real-world, such as the cameras. Although we could try simulating the real-world using additional devices, for example, placing a screen in the front of the camera and controlling it from the testing software, this does not seem to work for ABVE, because there would still be variables outside the control of the binaries, such as the propagation of light between the screen and camera and image capture inside the camera. An intermediate approach could be to virtualize some of the vehicle's hardware and represent it as software. For example, the vehicle's cameras could be virtualized as software components, which could then be tested under an ABVE approach. The ABVE tool could then be used, generating inputs for the virtual cameras and trying to find NN robustness measure-related vulnerabilities.

The set of vulnerabilities to be automatically tested would grow, as new vulnerabilities were found, both in the real-world and as well as in new theoretical research. Compliance to safety properties imposed by regulations could also be added. Finally, Software Engineering could benefit from considering including automated NN robustess evaluation as a step of a software development life cycle [25], both in the context unit testing, as well as in the context of integration testing [8].

4 Acknowledgments

This research was supported by FCT-Portugal grant SFRH/BD/132560/2017

References

[1] Nick Stephens, John Grosen, Christopher Salls, Andrew Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Driller: Augmenting fuzzing through selective symbolic execution. In *NDSS*, volume 16, pages 1–16, 2016.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[3] Cormac Herley and PC van Oorschot. Science of security: Combining theory and measurement to reflect the observable. *IEEE Symposium on Security and Privacy*, 16(1):12–22, 2018.

[4] Sanjay Rawat, Vivek Jain, Ashish Kumar, Lucian Cojocar, Cristiano Giuffrida, and Herbert Bos. Vuzzer: Application-aware evolutionary fuzzing. In *Network and Distributed System Security Symposium*, 2017.

[5] Van-Thuan Pham, Marcel Bhme, and Abhik Roychoudhury. Model-based whitebox fuzzing for program binaries. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 543–553, 2016.

[6] Marcel Bhme, Van-Thuan Pham, and Abhik Roychoudhury. Coverage-based greybox fuzzing as markov chain. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1032–1043, 2016.

[7] Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Andrew Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, and Giovanni Vigna. Sok: (state of) the art of war: Offensive techniques in binary analysis. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 138–157, 2016.

[8] Mark Fewster and Dorothy Graham. *Software Test Automation*. Addison-Wesley Professional, 1999. ISBN 0201331403.

[9] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[10] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[11] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference on*, pages 278–287, 2017.

[12] Hossein Hosseini, Baicen Xiao, and Radha Poovendran. Google's cloud vision api is not robust to noise. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 101–105, 2017.

[13] W. E. Burr, D. F. Dodson, and W. T. Polk. Nist sp 800-63: Electronic authentication guideline (version 1.0) - password guidelines. Technical report, 2004.

[14] Paul A. Grassi. Nist sp 800-63b: Digital identity guidelines - authentication and lifecycle. Technical report, 2017.

[15] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, 2012.

[16] Matt Weir, Sudhir Aggarwal, Michael P. Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 162–175, 2010.

[17] Yinqian Zhang, Fabian Monrose, and Michael K. Reiter. The security of modern password expiration: an algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 176–186, 2010.

[18] David L. Goodstein. *On Fact and Fraud: Cautionary Tales from the Front Lines of Science*. Princeton University Press, 2010. ISBN 978-0691139661.

[19] Philippe Gorry. Medical literature imprinting by pharma ghost writing: A scientometric evaluation. In *ISSI*, 2015.

[20] Herbert A. Simon. Rational choice and the structure of the environment. *Psychological Review, Vol*, 63(2):129–138, 1956. doi: 10.1037/h0042769.

[21] Steve Keen. *Debunking Economics - Revised and Expanded Edition: The Naked Emperor Dethroned?* Zed Books, 2011. ISBN 978-1848139923.

[22] Reinhard Sippel. An experiment on the pure theory of consumer's behaviour. *The Economic Journal*, 107(444):1431–1444, 1997.

[23] J. Doyne Farmer. Economics needs to treat the economy as a complex system. *Complexity Research Initiative for Systemic instabilities (CRISIS)*, 2012.

[24] Gim Hee Lee, Friedrich Faundorfer, and Marc Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2746–2753, 2013.

[25] Gaurav Kumar and Pradeep Kumar Bhatia. Comparative analysis of software engineering models from traditional to modern methodologies. In *2014 Fourth International Conference on Advanced Computing and Communication Technologies*, pages 189–196, 2014.