FoMSESS Jahrestagung 2022

Extended Abstracts

Die Fachgruppe FoMSESS[1] im GI-Fachbereich Sicherheit beschäftigt sich mit der Anwendung von Formalen Methoden und Software Engineering in der Entwicklung sicherer Systeme.

In ihren Jahrestreffen bietet die Fachgruppe die Möglichkeit, über aktuelle Forschungsarbeiten zu berichten und zu diskutieren und sich mit Gleichgesinnten zu vernetzen.

Nach den Online-Jahrestreffen 2020 und 2021 wurde auch das Jahrestreffen 2022 online durchgeführt. Auch diesmal gelang es, zwei Nachmittage mit interessanten Vorträgen und lebhaften Diskussionen zu füllen. Die Vortragenden bekamen auch dieses Jahr die Möglichkeit, Extended Abstracts ihrer Beiträge zu verfassen, um diese auf der FoMSESS-Webseite zu veröffentlichen. Das Ergebnis sehen Sie gerade vor sich.

Viel Spaß beim Lesen!

Zoltan Mann, November 2022

[1] https://fg-fomsess.gi.de

A Manifesto for Applicable Formal Methods – Extended Abstract

Mario Gleirscher, Universität Bremen

October 12, 2022

The successful use of formal methods in critical systems practice is far from many researchers' and practitioners' hopes and expectations. This observation seems alarming inasmuch as formal methods have long been advocated as one of the best paradigms to construct or assess dependable systems. One could thus expect these methods to be in wide-spread use for engineering critical systems. Unsurprisingly, the formal methods and software engineering communities have led a regular debate over the past four decades about the various causes for the observed lack of formal methods transfer and use.

In a literature survey combined with a SWOT analysis—an analysis of Strengths, Weaknesses, Opportunities, and Threats—, we summarise research on the potential causes and highlight key opportunities for employing the formal methods paradigm to the maximum benefit, for example, by increasing the level of method integration (Gleirscher, Foster, et al. 2019). Our practitioner survey (Gleirscher and Marmsoler 2020) extends this assessment of the phenomenon by two key findings from self-reported empirical data. Firstly, the survey respondents—to a large extent formal methods users from industry—indicate an intent to increase their use of these methods. Secondly, however, our respondents also suggest that the applicability of these methods, whether or not by using the corresponding tools, is still insufficient for them to unfold their full potential. The respondents confirm known and identify new obstacles to the transfer and use of formal methods. Our results suggest new research directions.

Based on our assessments, we revisited the concept of *applicable formal methods*, organised a workshop[1] to revitalise the currently rather scattered community of inclined researchers, and developed a proposal in the form of a *Manifesto for Applicable Formal Methods* (Gleirscher, van de Pol, et al. 2021). We hope that the manifesto fosters the development of a worldwide community of researchers and practitioners devoted to enhancing the available methods and tools in support of the *principles* proposed and *impacts* envisaged by the manifesto and, ultimately, to intensify the use of these methods.

In a recent working paper (Gleirscher, Sachtleben, et al. 2022), we evaluate the state of the art of *tool qualification for formal methods*, that is, approaches to increasing the dependability and *explainability*—one of the manifesto's principles—of tools supporting the application of certain formal methods, such as proof assistants, model checkers, and certified compilers. We develop a position about the next steps in this direction, focusing on what is known as *artefact-based tool qualification*.

References

Gleirscher, Mario, Simon Foster, and Jim Woodcock (2019). "New Opportunities for Integrated Formal Methods". In: *ACM Comput. Surv.* 52 (6), 117:1–117:36. DOI: `10.1145/3357231`. arXiv: `1812.10103 [cs.SE]`.

Gleirscher, Mario and Diego Marmsoler (2020). "Formal Methods in Dependable Systems Engineering: A Survey of Professionals from Europe and North America". In: *Empir Softw Eng* 25.6, pp. 4473–4546. DOI: `10.1007/s10664-020-09836-5`. arXiv: `1812.08815 [cs.SE]`.

Gleirscher, Mario, Jaco van de Pol, and James Woodcock (2021). *A Manifesto for Applicable Formal Methods*. Working paper. University of Bremen, Aarhus University, and University of York. arXiv: `2112.12758 [cs.SE]`.

Gleirscher, Mario, Robert Sachtleben, and Jan Peleska (2022). *Qualification of Proof Assistants, Checkers, and Generators: Where We Are and What Next?* Working paper. University of Bremen.

[1] See `https://sites.google.com/view/appfm21/home`.

Algorithms for Verification of Neural Networks

Philipp Kern

Karlsruhe Institute of Technology (KIT), Germany
philipp.kern@kit.edu

Abstract. Neural networks are increasingly applied in safety critical domains like self-driving cars and collision avoidance for aircraft, where errors can have severe consequences. This necessitates provable guarantees about their behaviour and has made the development of dedicated verification algorithms an active area of research. I characterize situations where machine learning is usually applied and talk about what can be verified under these circumstances. This is followed by a brief overview of different proposed verification algorithms, before I focus on one successful method - symbolic interval propagation. Furthermore, I present various optimizations that help to increase verification performance.

1 Introduction

Nowadays neural networks (NNs) achieve state-of-the-art performance for many machine learning tasks such as mastering the games of chess and go [Si18a] or image recognition [Yu22]. Their successes have led to the application of NNs in sensitive and even safety-critical domains like crime prediction [BDE09] and autonomous driving [Bo16] or airborne collision avoidance [Ju16].

In order to provide much needed guarantees about the behaviour of NNs, researchers in the field of NN verification have developed powerful techniques to formally prove or disprove various properties of trained NNs.

In this extended abstract, we first present a characterization of general scenarios where NNs or machine learning in general are applied and what properties can be verified in these cases. Then we continue with a brief overview of existing methods for NN verification, before we specifically concentrate on *symbolic interval propagation* and various optimizations that we proposed in [KBS22] to enhance the performance of this technique.

2 Use Cases and Applicable Verification Properties

In some use cases such as image recognition, a mathematical formulation of the desired property (e.g. this image represents a stop sign) is at very complicated, if not impossible to define. This lack of a closed-form formal model often is even the main reason, why machine learning methods are applied to a specific problem. While a way to specify the exact property one wants to verify (e.g. that every image of a stop sign is correctly classified as a stop sign) might not be known, it is still possible to verify necessary conditions [Eh19]. In the context of image recognition for example *adversarial robustness* [Sz13]

$$\forall \mathbf{x} : \|\mathbf{x}_0 - \mathbf{x}\| \leq \varepsilon \wedge NN(\mathbf{x}_0) = y \Rightarrow NN(\mathbf{x}) = y \tag{1}$$

specifies that all images in an ε-neighbourhood (with ε so small that humans will not notice a difference to $\mathbf{x}_0$) around some initial image $\mathbf{x}_0$ need to be classified as belonging to the same class y as $\mathbf{x}_0$.

When NNs are used to replace existing algorithms in order to achieve faster runtime or smaller memory consumption [Ju16], one can simply check the NN against existing specification of the replaced algorithm [Eh19].

Large NNs themselves, however, might also be expensive to execute or require large amounts of memory prohibiting their use on edge devices like smart phones or embedded systems. To solve this problem, several methods have been proposed to obtain a reduced-size NN $\mathcal{R}$ from a larger NN $\mathcal{N}$ without sacrificing too much performance. A useful property in these cases is the verification of ε-equivalence [PWW20,Pa20,KBKS20,TKBS21]: Proving that the outputs of $\mathcal{R}$ and $\mathcal{N}$ only differ by at most some small constant ε over inputs in a given region $\mathbf{x} \in \mathcal{X}$. By proving *one-hot equivalence* [KBKS20], we can also verify whether the outputs of $\mathcal{R}$ and $\mathcal{N}$ correspond to the same classification result.

Concerning the application of NNs in sensitive domains like prediction of credit scores or crime, the authors of [KS22] for example focus on verifying *global individual fairness* to ensure that the NN does not discriminate against minority groups based on protected attributes like race or gender.

3 Verification Techniques

All properties presented in the previous section can be written in the typical form of a NN verification problem, which can be phrased as the satisfiability problem

$$\exists \mathbf{x}, \mathbf{y}. \mathbf{x} \in \mathcal{P} \wedge \mathbf{y} = NN(\mathbf{x}) \wedge \mathbf{y} \in \mathcal{Q} \ , \tag{2}$$

where $\mathcal{P}$ and $\mathcal{Q}$ are polytopes. This problem is decidable for feed-forward ReLU-NNs (and δ-decidable for feed-forward NNs with sigmoidal activation functions [Iv19]), but unfortunately the problem is also NP-complete [Ka17].

Nevertheless, efficient algorithms for the solution of the problem have been proposed in recent years – also stimulated by a yearly competition [BLJ21]. We can roughly classify them into methods primarily based on optimization and methods primarily based on abstract interpretation although most algorithms make use of both techniques to a different extent.

Methods using abstract interpretation propagate an input set $\mathcal{X}$ through a NN using different abstract domains like zonotopes [Si18b,St21], convex starsets [Ba20] or symbolic intervals [Wa18b,Wa18a,Si19] to obtain an overapproximation $\mathcal{Y} \supseteq NN(\mathcal{X}$ of the reachable output set. If the emptiness of the intersection of a forbidden set $\mathcal{F}$ and $\mathcal{Y}$ can be determined, the property is guaranteed to hold, whereas an input point $\mathbf{x} \in \mathcal{X}$ for which $NN(\mathbf{x}) \in \mathcal{F}$ constitutes a counterexample.

Optimization-based approaches typically maximize the violation $v_C(\mathbf{x})$ of a constraint C for inputs $\mathbf{x} \in \mathcal{X}$ of the NN using for example mixed integer linear programming (MILP) [CNR17,KBKS20] or semidefinite programming [Da20]. If a bound on the maximal value of $v_C^* = \max_{x \in \mathcal{X}} v_C(\mathbf{x}) \leq 0$ can be found, the constraint is guaranteed to hold. On the other hand, any input $\mathbf{x} \in \mathcal{X}$ with $v_C(\mathbf{x}) > 0$ constitutes a counterexample to the validity of the constraint.

Usually both abstract interpretation- and optimization-based approaches can efficiently handle the affine operations in NNs like matrix multiplications or propagation through fixed-active or fixed-inactive ReLU functions, but loose precision or require refinement steps for unstable ReLUs, if the current property could neither be verified nor disproven. The size of the NN as well as the size of the input domain have a large influence on the number of unstable ReLUs, and therefore on required verification time.

3.1 Symbolic Interval Propagation

Symbolic interval propagation is an abstract-interpretation technique that propagates a hyperrectangular input set $\mathcal{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] \subseteq \mathbb{R}^n$ through a NN. In contrast to traditional interval arithmetic, the reachable sets for intermediate neurons n_i are not represented by intervals of concrete numbers $n_i \in [\underline{n_i}, \overline{n_i}]$, $\underline{n_i}, \overline{n_i} \in \mathbb{R}$, but by symbolic intervals of affine functions $[l(\mathbf{x}), u(\mathbf{x})]$ in the input variables such that $l(\mathbf{x}) = l_0 + \sum_{i=1}^n l_i x_i, u(\mathbf{x}) = u_0 + \sum_{i=1}^n u_i x_i$ and $l(\mathbf{x}) \leq n_i \leq u(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. A concrete upper bound $\overline{n_i}$ for n_i can be obtained by substituting the lower bounds $\underline{x_i}$ for inputs with negative coefficients in $u(\mathbf{x})$ and the upper bounds upx_i for inputs with positive coefficients in $u(\mathbf{x})$. For a concrete lower bound, one needs to substitute the appropriate input bounds according to the coefficients in $l(\mathbf{x})$. Since each input x_i only appears once $u(\mathbf{x})$ (and $l(\mathbf{x})$), we never use both $\underline{x_i}$ and $\overline{x_i}$ simultaneously, when computing a concrete upper bound (or lower bound), which alleviates the dependency issue with respect to the input variables compared to traditional interval arithmetic.

However, we might still implicitly use both the symbolic lower and upper bound $l(\mathbf{x})$ and $u(\mathbf{x})$ of an intermediate neuron n_i, when computing the symbolic bounds of later neurons. While there are methods that alleviate this problem [Si18b,Si19,Zh18], they either restricted to parallel relaxations of unstable ReLU-neurons or require a quadratic number of backsubstitution passes. Symbolic interval propagation with the introduction of fresh variables for unstable neurons [Pa20] on the other hand, also reduces the dependency problem, while still allowing for non-parallel ReLU relaxations and only requiring a single symbolic forward pass. In order to avoid multiple backsubstitutions, however, the bounds of the introduced fresh variables need to be stored in terms of the input variables, which leads to a loss of symbolic information, if too many fresh variables are introduced.

In our work [KBS22], we therefore only introduce a fixed number of fresh variables and for at most a fraction of λ of the neurons of each layer of an NN. Furthermore, we prioritize introduction of fresh variables for neurons whose concrete lower and upper bounds are far apart, with the intuition that implicitly utilizing both the symbolic lower and upper bounds of these neurons would yield worse bounds later on than utilizing both bounds, when they are only slightly different. Additionally, we utilize information about the influence of an input x_i on the bounds of unstable intermediate neurons to guide our refinement process based on branch-and-bound on the input domain. Instead of just splitting the input x_i of the NN with the largest range $\overline{x_i} - \underline{x_i}$, we obtain a splitting score by multiplying the range with the sum of the absolute values of the coefficients of x_i in the lower and upper symbolic bounds $l(\mathbf{x})$ and $u(\mathbf{x})$ of unstable neurons and bisect the input with the largest splitting score. Similarly, we also use information from the symbolic bounds for faster counterexample generation. Instead of just sampling the center point of the current input hyperrectangle, we calculate the symbolic upper bound on the violation of the property to prove and use the input that maximizes this symbolic upper bound as a counterexample-candidate. Our method performs well especially on medium sized neural networks with low dimensional input space. Therefore we evaluated our implementation DP-NeurifyFV[1] of this optimized approach on the popular ACAS Xu benchmark suite [Ju16] containing 45 networks with 5 inputs and 300 neurons each and showed significant runtime improvements compared to the state-of-the-art tools ZoPE[St21] and nnenum[Ba20].

4 Conclusion

Nowadays, NNs are already applied in safety-critical and sensitive areas like automated driving or credit score prediction. Therefore provable guarantees about the behaviour of such NNs are necessary. Even if we do not have full specifications of the desired properties, useful properties can still be formulated. This includes necessary conditions, partial specifications, fairness properties and equivalence between two NNs. Furthermore the problem of NN verification is decidable for feed-forward NNs with ReLU activation functions and we have seen several algorithms that are capable of verifying small to medium-sized NNs dependent on the property to verify (especially considering the size of the input set).

However, the problem of NN verification is far from being completely solved. As NNs become ever larger, scalability to those large models constitutes a major challenge to verification algorithms. Furthermore, although we have seen that useful properties can be formulated, we still cannot accurately capture many of the properties that we actually would like to verify. Major research effort is required to make progress in that direction.

References

Ba20. Bak, Stanley; Tran, Hoang-Dung; Hobbs, Kerianne; Johnson, Taylor T.: Improved Geometric Path Enumeration for Verifying ReLU Neural Networks. In: CAV (1). volume 12224 of Lecture Notes in Computer Science. Springer, pp. 66–96, 2020.

[1] available at `https://github.com/phK3/DPNeurifyFV.jl`

BDE09. Brennan, Tim; Dieterich, William; Ehret, Beate: Evaluating the Predictive Validity of the Compas Risk and Needs Assessment System. Criminal Justice and Behavior, 36:21 – 40, 2009.

BLJ21. Bak, Stanley; Liu, Changliu; Johnson, Taylor: , The Second International Verification of Neural Networks Competition (VNN-COMP 2021): Summary and Results, 2021.

Bo16. Bojarski, Mariusz; Testa, Davide Del; Dworakowski, Daniel; Firner, Bernhard; Flepp, Beat; Goyal, Prasoon; Jackel, Lawrence D.; Monfort, Mathew; Muller, Urs; Zhang, Jiakai; Zhang, Xin; Zhao, Jake; Zieba, Karol: , End to End Learning for Self-Driving Cars, 2016.

CNR17. Cheng, Chih-Hong; Nührenberg, Georg; Ruess, Harald: Maximum Resilience of Artificial Neural Networks. Lecture Notes in Computer Science, p. 251–268, 2017.

Da20. Dathathri, Sumanth; Dvijotham, Krishnamurthy; Kurakin, Alexey; Raghunathan, Aditi; Uesato, Jonathan; Bunel, Rudy; Shankar, Shreya; Steinhardt, Jacob; Goodfellow, Ian J.; Liang, Percy; Kohli, Pushmeet: Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In: NeurIPS. 2020.

Eh19. Ehlers, Rüdiger: , Safety and Reliability for Learning Systems, 2019.

Iv19. Ivanov, Radoslav; Weimer, James; Alur, Rajeev; Pappas, George J.; Lee, Insup: Verisig: verifying safety properties of hybrid systems with neural network controllers. Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, 2019.

Ju16. Julian, Kyle D; Lopez, Jessica; Brush, Jeffrey S; Owen, Michael P; Kochenderfer, Mykel J: Policy compression for aircraft collision avoidance systems. In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). IEEE, pp. 1–10, 2016.

Ka17. Katz, Guy; Barrett, Clark; Dill, David L.; Julian, Kyle; Kochenderfer, Mykel J.: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. Lecture Notes in Computer Science, p. 97–117, 2017.

KBKS20. Kleine Büning, Marko; Kern, Philipp; Sinz, Carsten: Verifying equivalence properties of neural networks with relu activation functions. In: International Conference on Principles and Practice of Constraint Programming (CP). Springer, pp. 868–884, 2020.

KBS22. Kern, Philipp; Büning, Marko Kleine; Sinz, Carsten: Optimized Symbolic Interval Propagation for Neural Network Verification. In: 1st Workshop on Formal Verification of Machine Learning (WFVML 2022) colocated with ICML 2022: International Conference on Machine Learning. July 2022.

KS22. Khedr, Haitham; Shoukry, Yasser: CertiFair: A Framework for Certified Global Fairness of Neural Networks. CoRR, abs/2205.09927, 2022.

Pa20. Paulsen, Brandon; Wang, Jingbo; Wang, Jiawei; Wang, Chao: NEURODIFF: Scalable Differential Verification of Neural Networks using Fine-Grained Approximation. In: 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia. IEEE, pp. 784–796, 2020.

PWW20. Paulsen, Brandon; Wang, Jingbo; Wang, Chao: ReluDiff: differential verification of deep neural networks. In (Rothermel, Gregg; Bae, Doo-Hwan, eds): ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea. ACM, pp. 714–726, 2020.

Si18a. Silver, David; Hubert, Thomas; Schrittwieser, Julian; Antonoglou, Ioannis; Lai, Matthew; Guez, Arthur; Lanctot, Marc; Sifre, Laurent; Kumaran, Dharshan; Graepel, Thore; Lillicrap, Timothy; Simonyan, Karen; Hassabis, Demis: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419):1140–1144, 2018.

Si18b. Singh, Gagandeep; Gehr, Timon; Mirman, Matthew; Püschel, Markus; Vechev, Martin T.: Fast and Effective Robustness Certification. In: NeurIPS. pp. 10825–10836, 2018.

Si19. Singh, Gagandeep; Gehr, Timon; Püschel, Markus; Vechev, Martin T.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang., 3(POPL):41:1–41:30, 2019.

St21. Strong, Christopher A.; Katz, Sydney M.; Corso, Anthony L.; Kochenderfer, Mykel J.: ZoPE: A Fast Optimizer for ReLU Networks with Low-Dimensional Inputs. 2021.

Sz13. Szegedy, Christian; Zaremba, Wojciech; Sutskever, Ilya; Bruna, Joan; Erhan, Dumitru; Goodfellow, Ian; Fergus, Rob: , Intriguing properties of neural networks, 2013.

TKBS21. Teuber, Samuel; Kleine Büning, Marko; Sinz, Carsten: Geometric Path Enumeration for Equivalence Verification of Neural Networks. In: International Conference on Tools with Artificial Intelligence (ICTAI). 2021.

Wa18a. Wang, Shiqi; Pei, Kexin; Whitehouse, Justin; Yang, Junfeng; Jana, Suman: Efficient Formal Safety Analysis of Neural Networks. In: NeurIPS. pp. 6369–6379, 2018.

Wa18b. Wang, Shiqi; Pei, Kexin; Whitehouse, Justin; Yang, Junfeng; Jana, Suman: Formal Security Analysis of Neural Networks using Symbolic Intervals. In: USENIX Security Symposium. USENIX Association, pp. 1599–1614, 2018.

Yu22. Yu, Jiahui; Wang, Zirui; Vasudevan, Vijay; Yeung, Legg; Seyedhosseini, Mojtaba; Wu, Yonghui: CoCa: Contrastive Captioners are Image-Text Foundation Models. ArXiv, abs/2205.01917, 2022.

Zh18. Zhang, Huan; Weng, Tsui-Wei; Chen, Pin-Yu; Hsieh, Cho-Jui; Daniel, Luca: Efficient Neural Network Robustness Certification with General Activation Functions. In: NeurIPS. pp. 4944–4953, 2018.

Automatic online quantification and prioritization of data protection risks

Extended Abstract

Sascha Sven Zmiewski[1], Jan Laufer[1], and Zoltán Ádám Mann[2]

[1]University of Duisburg-Essen, Essen, Germany
[2]University of Amsterdam, Amsterdam, Netherlands

October 30, 2022

Abstract

Data processing systems operate in increasingly dynamic environments. In such environments, changes at run time can result in the dynamic appearance of data protection vulnerabilities. An autonomous system can mitigate such vulnerabilities by means of automated self-adaptations. If there are several data protection vulnerabilities at the same time, the system has to decide which ones to address first. We propose an approach to automatically quantify and prioritize data protection vulnerabilities at run time. Full details on our approach including, e.g., the mathematical model, and the experiments performed can be found in [ZLM22]. The following summary presents the motivation behind our approach, our the main idea, and selected evaluation results.

1 Introduction

An increasing number of software applications process personal data, raising concerns about data protection [Rub13, And14]. This is mirrored by recent data protection regulation, such as the General Data Protection Regulation (GDPR) of the European Union (EU). Data protection covers several aspects of security and privacy. In particular, a key data protection requirement is that personal data should only be accessible by authorized actors [TMM+20].

Software can be made reasonably secure *by design* for a known and stable environment. However, data processing systems, increasingly often, operate in increasingly dynamic environments, such as in cloud or edge computing. In such environments, changes at run time can result in the dynamic appearance of data protection vulnerabilities, i.e., configurations in which an attacker could gain unauthorized access to confidential data. An autonomous system can mitigate such vulnerabilities by means of automated self-adaptations (see e.g., [MKL+21]. If there are several vulnerabilities at the same time, the system has to decide which ones to address first.

In other areas of cybersecurity, risk-based approaches have proven useful for prioritizing where to focus efforts for increasing security. Traditionally, risk assessment is a manual and time-consuming process. On the other hand, addressing run-time risks requires timely decision-making, which in turn necessitates automated risk assessment.

We propose an approach for automatically quantifying the risks associated with data protection vulnerabilities arising at run time. Our approach is based on a new mathematical model that formalizes the key properties of data protection vulnerabilities: the dependence of the caused damage on the type and amount of the involved data and on the time that elapsed since the vulnerability arose. This model allows us to automatically compute the risk values of vulnerabilities, and thus to prioritize the vulnerabilities based on risk. We implement our approach as an extension to the RADAR self-adaptive data protection system [MKL+21].

2 Context

The research presented in this paper is performed in the context of RADAR, an approach for ensuring data protection in systems that are exposed to changes at run time [MKL+21]. RADAR automatically identifies data protection vulnerabilities at run time and determines the best adaptations to keep data protected. A *data protection vulnerability* is a configuration of the system and its environment, in which an attacker may gain unauthorized access to some confidential data. RADAR serves as an example system to motivate and validate our approach. However, our approach could also be integrated into other systems that detect and mitigate data protection vulnerabilities.

RADAR uses an adaptation planning algorithm to identify the best adaptation. The algorithm needs to take into account that there may be multiple vulnerabilities, there may be multiple adaptations to mitigate a vulnerability, and an adaptation to mitigate a vulnerability may also mitigate or create other vulnerabilities. A sequence of adaptations may be needed to mitigate all vulnerabilities. In this sequence, the order of the adaptations may be important because an adaptation may become applicable only after another adaptation was carried out.

Searching for the best adaptation sequence can be time-consuming. While RADAR is searching, an attacker could already exploit the vulnerabilities. Therefore, the search is only allowed for a limited time (10 seconds in [MKL+21]). RADAR tries to mitigate as many vulnerabilities as possible in the given time frame. However, RADAR is lacking the intelligence to know which vulnerabilities are more important or more urgent than others.

3 Proposed Approach

The aim of our approach is to quantify the risk associated with data protection vulnerabilities in a system. In the following, we summarize the idea behind our approach. More details can be found in [ZLM22].

A data protection vulnerability is a special kind of security vulnerability, with some typical properties. First, the damage caused by an attack depends on the type of data (e.g., in a hospital, patients' health data is more sensitive than inventory data) and on the amount of data (the more data is stolen, the higher the damage). Second, exploiting a data protection vulnerability takes time. This includes time to detect the existence of the vulnerability, time to prepare an attack, and time to steal each data item. The time for stealing the whole data set depends on the size of the data set. Third, once the attacker has stolen the data, the damage cannot be undone.

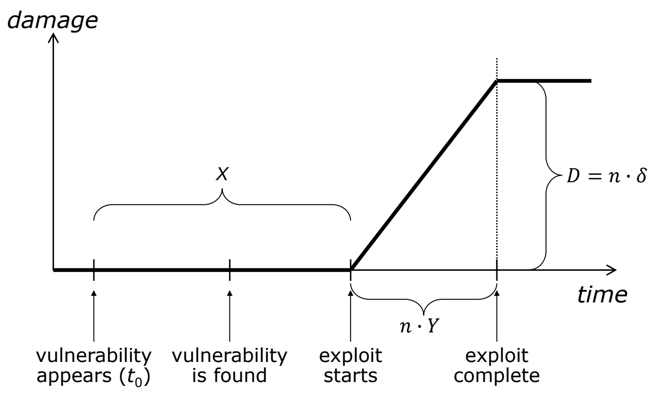

Figure 1: Damage caused by an attack

To account for these properties, Fig. 1 sketches our model of how the caused damage changes with time. The model can be seen as the impact of the cyber kill chain [YR15] on data protection. From the moment that a vulnerability appears, it takes the attacker some time to detect the vulnerability and then to prepare an exploit. The time to detect the vulnerability and to prepare the exploit may be very short if the vulnerability is easily detected / exploited, but it may also be long for more intricate vulnerabilities. These preparatory steps of the attacker do not cause damage. During the exploit, the damage increases linearly with time, as proportionately more and more data is stolen. When all data is stolen, the damage reaches its maximum, and it stays at that level.

Our attack model is primarily aimed at modeling attacks against the confidentiality of data. Other types of attacks, for example against the availability of data (e.g., ransomware attacks) could also follow a similar pattern, and could thus be amenable to similar reasoning, but we leave this for future research.

Our approach is not to wait until an attack actually happens. As soon as the vulnerability appears, the defender can start reasoning about possible attacks to exploit the vulnerability, making it possible to proactively mitigate the risk of such an attack. The defender can use the general model of Fig. 1, but without knowing the exact parameters (e.g., how long the attacker will need to detect the vulnerability). Rather, the defender can perform a probabilistic analysis, as described next, to reason about the risks.

Let us consider a data protection vulnerability, i.e., a vulnerability that potentially allows an attacker to gain access to a confidential dataset containing n items. The maximum damage that an attacker can cause is $D = n \cdot \delta$, corresponding to the case when the attacker manages to steal the whole dataset. Here, $\delta > 0$ is a given constant, corresponding to the damage caused by stealing one data item, and thus defining how valuable the data in the given dataset is.

Let t_0 denote the point in time when the vulnerability appears. Let the random variable X denote the time it takes an attacker to start exploiting the vulnerability. (In Fig. 1, X is the total time for noticing the vulnerability and preparing the exploit.) Let the random variable Y denote the time it takes the attacker, after the exploit started, to steal one data item. Let t denote the time that elapsed since t_0. The damage caused by the attacker during this time is, as can be seen from Fig. 1:

$$d_n(t) = \begin{cases} 0 & \text{if } t < X \\ \frac{t-X}{Y} \cdot \delta & \text{if } X \leq t < X + n \cdot Y \\ D & \text{if } t \geq X + n \cdot Y \end{cases} \tag{1}$$

Note that, since X and Y are random variables, so is $d_n(t)$.

By focusing on the expected damage in the next period of time, we obtain the right metric for risk prioritization. We disregard the already incurred damage, which is a sunken cost, and thus should not influence the decision-making. We also disregard damage that will be incurred in the far future because that damage can be avoided regardless of how current risks are prioritized.

To prioritize vulnerabiltiies we propose two algorithms: The first one is using Monte Carlo sampling for different n and t values in a preprocessing step, the second one is used to quantify the risk of a vulnerability at run time. Note that run-time changes of n or t may lead to a new risk value for the same vulnerability.

4 Evaluation

We experimentally evaluate the integration of our method into RADAR and compare three different prioritization methods. These are NoPrioritiziation (NoPrio for short) since all vulnerabilities are considered equal, LocalPrioritization (LocalPrio), and GlobalPriotization (GlobalPrio). LocalPrio is to eliminate the vulnerabilities with the highest risk value first, and then continuing in descending order of risk value. The idea of GlobalPrio is to minimize the sum of the risk values of the vulnerabilities in the system configuration.

For the experiment, we use a system representation based on a project partner's real cloud system, which was multiplied 21 times around a central object [MKL+21]. This initial run-time model does not contain any vulnerabilities. In a controlled scenario multiple vulnerabilities are added to the system be performing system modifications. In some experiments, all vulnerabilities are injected at once. In others, the injection of vulnerabilities occurs in two phases to simulate the coexistence of older and newer vulnerabilities.

In a first experiment, we show how RADAR prioritizes the mitigation of different vulnerabilities. Newer vulnerabilities are prioritized over older ones. The older vulnerabilities are already almost completely exploited, which results in lower potential damage for the future and thus in a lower risk value than the newer ones, which are associated with higher potential future damage[1].

Fig. 2 compares the results for the different prioritization methods in a boxplot. We group the vulnerabilities by δ and report the total damage for these groups. In addition, the total damage for all vulnerabilities is also shown. The jagged line between 5,000,000 and 20,000,000 shows a break in the values of the vertical axis, as no damage values in this range were recorded and thus this range was cut out to increase readability. Further experiments show the respective impact of the important parameters δ, dataset size, and older vs. newer vulnerabilities.

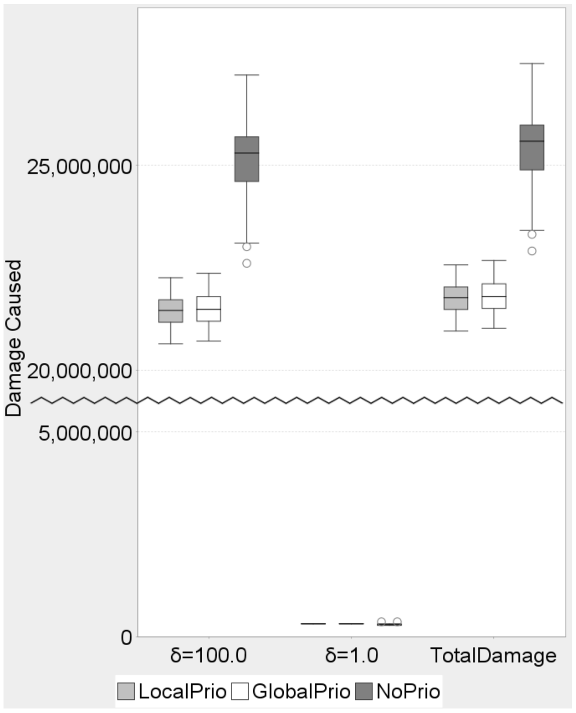

Figure 2: Results for 100 runs of the experiment

All in all, GlobalPrio and LocalPrio lead to similar results while NoPrio performs significantly worse. Our approach allows the prioritization of risks, resulting in up to 15.8% reduction in damage.

References

[And14] Mark Andrejevic. The big data divide. *International Journal of Communication*, 8:1673–1689, 2014.

[MKL+21] Zoltán Ádám Mann, Florian Kunz, Jan Laufer, Julian Bellendorf, Andreas Metzger, and Klaus Pohl. RADAR: Data protection in cloud-based computer systems at run time. *IEEE Access*, 9:70816–70842, 2021.

[Rub13] Ira Rubinstein. Big data: the end of privacy or a new beginning? *International Data Privacy Law*, 2013.

[TMM+20] Aggeliki Tsohou, Emmanouil Magkos, Haralambos Mouratidis, George Chrysoloras, Luca Piras, Michalis Pavlidis, Julien Debussche, Marco Rotoloni, and Beatriz Gallego-Nicasio Crespo. Privacy, security, legal and technology acceptance elicited and consolidated requirements for a GDPR compliance platform. *Information and Computer Security*, 28(4):531–553, 2020.

[YR15] Tarun Yadav and Arvind Mallari Rao. Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication*, pages 438–452. Springer, 2015.

[ZLM22] Sascha Sven Zmiewski, Jan Laufer, and Zoltán Ádám Mann. Automatic online quantification and prioritization of data protection risks. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–10, 2022.

[1] This holds true for the given values of the parameters. For other values, it is also possible that older vulnerabilities are to be prioritized over newer ones. For example, if the newer vulnerabilities are not yet expected to lead to damage in the next time period, while older vulnerabilities already start to lead to damage, then older vulnerabilities are prioritized.

Helping Developers with Password Security

Alena Naiakshina
Ruhr University Bochum
Germany

October 12, 2022

Abstract

Authentication systems are a major concern for the usable security and privacy community. Twenty years ago, the seminal work "Users Are Not The Enemy" [1] by Adams and Sasse initiated a user-centred approach to research. This work was followed by extensive research on end users' security behavior around authentication systems, which resulted in many suggestions for improvement. Many of these proposals were passed on to software developers, who were considered experts who should know better and were expected to address the issues. However, the related work "Developers Are Not The Enemy" [2] by Green and Smith, citing Adams and Sasse [1], points out that, similar to end users, developers are usually not security experts. Thus, they also struggle with usability and security issues. This is highlighted by the high number of reported security breaches that have compromised millions of end-user passwords. In fact, much of the work invested in usable authentication systems might be in vain if software developers fail to securely store user passwords in databases.

Motivated by the recent security breaches, Naiakshina et al. [3, 4, 5, 6] provided deeper insights into developers' security behavior to clarify why software developers so often fail to store user passwords securely. For this, a Java-based password-storage study was conducted with different samples of developers: computer science (CS) students [3, 4], freelancers [5], and professional developers employed by different companies [6]. Participants were instructed to complete the registration functionalities for a social network platform. To investigate whether software developers think about security without prompting, half the participants were told the study was about application programming interface (API) usability (non-prompted for security), while the other half were specifically instructed to securely store user passwords (prompted for security). The study also investigated whether an API's level of password storage security support affects developers' security behavior. Thus, half the participants used a framework offering opt-in support for secure password storage (Spring), while the other half were provided a programming frame with JavaServer Faces (JSF), which required them to implement password storage security without support.

Initially, a qualitative and a quantitative study were conducted with 20/40 CS students from the University of Bonn in a laboratory setting [3, 4]. The most important finding of this study was that all the students who were not prompted for security submitted solutions in which user passwords were stored in plaintext in the database. Furthermore, a number of the prompted participants who considered password storage security still chose weak security practices. However, some participants claimed that they would have stored user passwords securely in the database if they were solving the task for a real company. In order to test whether these findings were a study artifact, a follow-up study was conducted with 43 freelancers recruited via Freelancer.com [5]. A pilot study with the freelancers suggested that a university context might lead them to believe that university students were hiring them to do their homework. Therefore, this time, the freelancers were not informed that the study was conducted by a research team. Instead, they were told that they were working for a start-up that had lost recently a developer from their team. In this study, freelancers behaved similarly to students with regard to user password storage. Freelancers also had often misconceptions about secure password storage and chose weak practices. Finally, 36 company developers were invited to take part in a password-storage study [6]. They were recruited through their companies and also via the German business social platform Xing. Company developers submitted significantly more secure solutions than students, and they also chose significantly better password storage parameters than students or freelancers. Thus, in absolute terms, they performed better than students and freelancers. However, in relative terms, the results were similar: Security prompting and framework had a significant effect on password storage security

for all samples. When prompted, more participants submitted secure solutions, and participants made better parameter choices for password storage security when they used Spring instead of JSF (freelancers used only JSF and thus were not tested for the variable framework).

Additionally, the four studies offered insights for the ecological validity of security studies with developers. The student studies provided an early indication that qualitative research might reveal essential insights without the need to conduct quantitative studies for specific use cases. Furthermore, if the usable security and privacy community is more interested in how security systems can be improved than in which developer group performs best, it might be valid to conduct studies with students rather than professionals. What is more, freelancers tended to behave similarly to students with regard to secure password storage, although they were not aware of the purpose of the study. This suggests that participants tended to ignore the security aspects of software even when working on a web application intended for the use in the real world.

A website provided by a trustworthy authority with state-of-the-art, ready-to-use security code might improve software security by taking the burden off developers. To aid developers in creating secure code for password-based authentication, Geierhaas et al. [7] developed and tested a programming resource, Let's Hash. Using Let's Hash, participants were between 5 and 32 times more likely to create secure code than those using their regular resources.

References

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Matthew Green and Matthew Smith. Developers are not the enemy!: The need for usable security apis. *IEEE Security & Privacy*, 14(5):40–46, 2016.

[3] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. Why do developers get password storage wrong?: A qualitative usability study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 311–328, New York, NY, USA, 2017. ACM.

[4] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. Deception task design in developer password studies: Exploring a student sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 297–313, Baltimore, MD, USA, 2018. USENIX Association.

[5] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, Emanuel von Zezschwitz, and Matthew Smith. "if you want, i can store the encrypted password": A password-storage field study with freelance developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 140:1–140:12, New York, NY, USA, 2019. ACM.

[6] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. On conducting security developer studies with cs students: Examining a password-storage study with cs students, freelancers, and company developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Lisa Geierhaas, Anna-Marie Ortloff, Matthew Smith, and Alena Naiakshina. Let's hash: Helping developers with password security. In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, pages 503–522, 2022.