**Tegawendé F. Bissyandé** is a chief scientist, associate professor at the University of Luxembourg, L-1359 Luxembourg, Luxembourg, where he conducts research on program debugging and repair at the Interdisciplinary Centre for Security, Reliability, and Repair. His research interests include program repair and software analytics. Bissyandé received a Ph.D. in computer science from the University of Bordeaux, France. He is a Member of IEEE. Contact him at tegawende.bissyande@uni.lu.

**Jacques Klein** is a full professor in software engineering and software security within the Interdisciplinary Centre for Security, Reliability, and Trust at the University of Luxembourg, L-1359 Luxembourg, Luxembourg. His main research interests are software security, software reliability, and data analytics. Klein received a Ph.D. in computer science from the University of Rennes, France. He is a Member of IEEE. Contact him at jacques.klein@uni.lu.

# Lessons Learned on Machine Learning for Computer Security

**Daniel Arp** | Technische Universität Berlin and University College London
**Erwin Quiring** | ICSI and Ruhr University Bochum
**Feargus Pendlebury** | University College London
**Alexander Warnecke** | Technische Universität Berlin
**Fabio Pierazzi** | King's College London
**Christian Wressnegger** | KASTEL Security Research Labs and Karlsruhe Institute of Technology
**Lorenzo Cavallaro** | University College London
**Konrad Rieck** | Technische Universität Berlin

**We identify 10 generic pitfalls that can affect the experimental outcome of AI driven solutions in computer security. We find that they are prevalent in the literature and provide recommendations for overcoming them in the future.**

Artificial intelligence (AI) and machine learning have enabled remarkable progress in science and industry. This advancement has naturally also impacted computer security, with nearly every major vendor now marketing AI-driven solutions for threat analysis and detection. Similarly, the number of research papers applying machine learning to solve security tasks has literally exploded.

These works come with the implicit promise that learning algorithms provide significant benefits compared with traditional solutions. In recent years, however, different studies have shown that learning-based approaches often fail to provide the promised performance in practice due to various restrictions ignored in the original publications.[1,2,3,4] In this article, we want to ask, Are there *generic* pitfalls that can affect the experimental outcome when applying machine learning in security? If so, how can researchers avoid stepping into them?

## Why Should I Care?

As a thorough researcher, one might tend to think, "This can

*Digital Object Identifier 10.1109/MSEC.2023.3287207*
*Date of current version: 11 September 2023*

never happen to me." However, as we will discuss in this article, pitfalls come in various forms and flavors, some obvious but others noticeable only with a very cautious eye. Hence, even experienced researchers might step into them from time to time without noticing. With this work, we want to raise awareness of these issues in the research community to reduce their prevalence in security research. Detailed recommendations and guidelines for each of the pitfalls can be found in the original paper.[5]

## A Motivating Example

To illustrate the problem, let us consider a learning-based method for the discovery of security vulnerabilities in source code as a motivating example.[6] As the manual auditing of source code is generally a time-consuming and tedious task, researchers have started to outsource it to machine learning algorithms. Here especially deep learning-based techniques have recently attained promising results in discovering vulnerabilities.[6,7,8]

Unfortunately, it has been shown that the performance of these models can often not keep up with the promises made.[8] Naturally, the question arises of what causes these huge discrepancies. In the following, we discuss some pitfalls that might have led to an overestimation of those methods' capabilities.

## Spurious Correlations

Even though deep neural networks have led to major breakthroughs in various areas, it is often unclear *why* they achieve this impressive performance. Fortunately, in recent years, several methods have been developed that enable the interpretation of these models, shedding some light on the decision-making process of neural networks.[9]

As an example, when analyzing a state-of-the-art method using these explanation techniques, we find that the highlighted features are barely connected to security vulnerabilities.[5,6] Instead, the most relevant features are meaningless tokens like brackets or commas in the source code, which have no semantic relevance to vulnerable code and thus represent noncausal *spurious correlations*. It seems as if these artifacts serve as shortcuts that allow the learning model to distinguish between vulnerable and nonvulnerable code. However, what could have caused this issue?

**However, as we will discuss in this article, pitfalls come in various forms and flavors, some obvious but others noticeable only with a very cautious eye.**

## Sampling Bias

A possible and common reason for the presence of spurious correlations is sampling bias. In this case, the distribution of the training data does not sufficiently represent the distribution at test time. Consequently, the model is not able to learn the underlying concept of the given task but rather relies on artifacts introduced in the training distribution. When composing a dataset for training the model, researchers need to be aware that there exist a variety of sources for sampling bias, some of which are very subtle. The strategy for collecting the data might thus bias the resulting dataset toward certain software versions, code authors, or programming languages.

## Data Snooping

Let us assume that the collected dataset does not suffer from sampling bias. Are there any other, less known, issues that might result in huge differences in the performance at training and test time? Indeed, another common pitfall that can lead to overoptimistic results is commonly referred to as *data snooping*. Here a learning model is trained with information that would not be available in practice. While this appears to be a pitfall that can be avoided very easily at the first glance, it turns out to be much harder to avoid than expected in many cases.

The reason is that data snooping exists in many different forms, some of which can be easily overlooked. For example, using incorrect time splits that ignore time dependencies within the data can inflate the actual performance.[1] Similarly, this pitfall applies if noisy data are removed from the test set based on knowledge that would normally not be available at training time. Even more subtle, solely evaluating well-known benchmark data might also overestimate the performance. Established benchmarks come with a history, so that researchers may unnoticeably use knowledge from prior work, including insights from the test distribution.

## The Greater Picture

The previous examples illustrate that there obviously exist a number of pitfalls that can harm the experimental outcome. However, the previously discussed issues are just the tip of the iceberg.

In this section, we provide a systematic overview of common pitfalls and explore their prevalence later in this article. To this end, we follow the individual stages of a typical machine learning pipeline. Figure 1 depicts the pipeline together with all pitfalls, and Table 1 provides a short description of each issue.

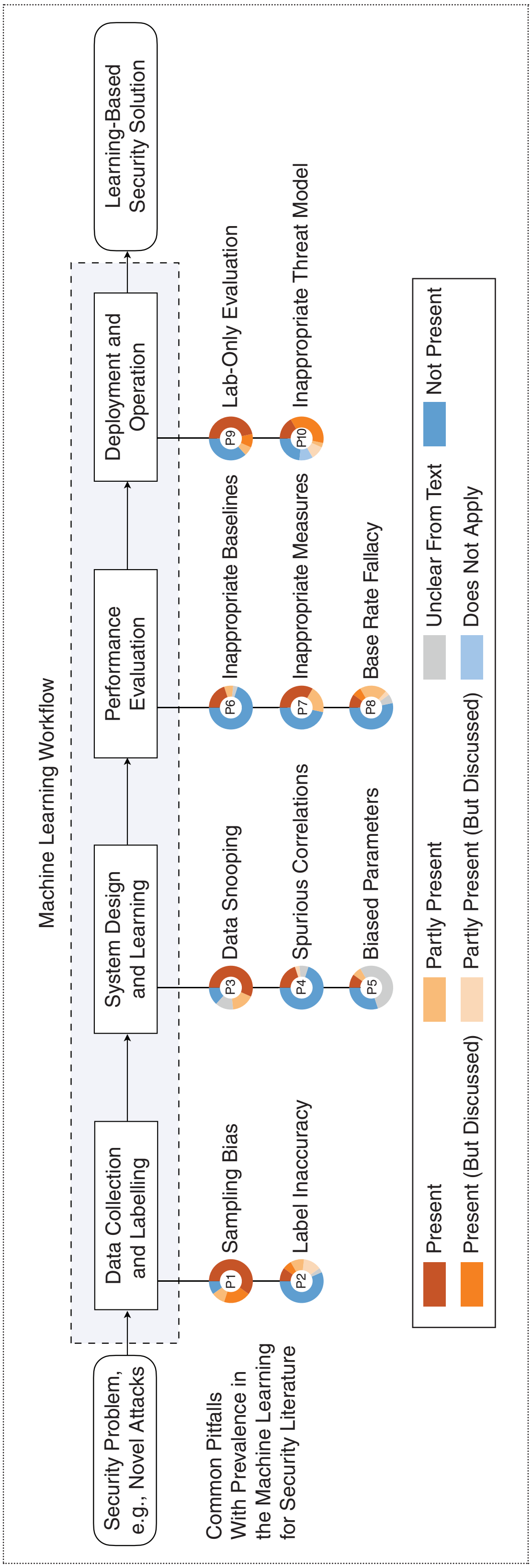


**Figure 1.** The common pitfalls of machine learning in computer security and their prevalence in the literature.

## Data Collection and Labeling Phase

Before we can start with developing a new learning-based method, we first have to collect an expressive dataset that resembles the data distribution we assume to see in practice. Moreover, we also often require meaningful label information if we want to apply supervised learning. Unfortunately, the composition of a realistic dataset with labels is often challenging, leading to the first two pitfalls: P1, sampling bias, and P2, label inaccuracy.

We have already discussed how sampling bias can affect the experimental outcome. Similarly, in the case of P2, the labels are erroneous or unstable, which, in turn, can also impact the performance of a learning model if we do not correct this noise.

## System Design and Learning Phase

Once we have composed a dataset, we can design and train our machine learning model. This stage includes the preprocessing of the data and the extraction of suitable features, as well as learning the actual model. In this stage, we can step in three different pitfalls when not being careful: P3, data snooping; P4, spurious correlations; and P5, biased parameter selection.

In the case of P3 and P5, the separation of training and test partition is flawed, so that the model uses information that is unavailable at test time, biasing the outcome of the experimental setup. For instance, the developers might ignore time dependencies within the collected data, such that the machine learning model is trained on data comprising future knowledge (which is not available outside the matrix). Another issue arises from P4. Here the feature design allows the model to pick up on artifacts unrelated to the security pattern, thus creating a shortcut for solving the actual tasks. While this can be unproblematic in

**Table 1. An overview of common pitfalls of machine learning in computer security.**

| | Pitfall | Description |
|---|---|---|
| P1 | Sampling bias | The composed dataset does not sufficiently represent the actual distribution. |
| P2 | Label inaccuracy | The ground-truth labels are inaccurate, unstable, or erroneous. |
| P3 | Data snooping | Information is used at training time that is usually not available in practice. |
| P4 | Spurious correlations | A learning model relies on false associations caused by artifacts unrelated to the task. |
| P5 | Biased parameter selection | Final parameters of a learning method are indirectly determined on the test set. |
| P6 | Inappropriate baseline | No adequate baseline methods are used in the evaluation for comparison. |
| P7 | Inappropriate performance measures | Used performance measures are not suitable for the application scenario. |
| P8 | Base rate fallacy | Large class imbalance is ignored when interpreting the performance. |
| P9 | Lab-only evaluation | The developed system is only tested in a laboratory setting. |
| P10 | Inappropriate threat model | Attacks against the machine learning component itself are not considered. |

some cases, it can also lead to serious problems and let the model fail completely during its deployment.

### Evaluation Phase

In the next stage, we evaluate the previously trained model and examine its performance on test data. Here we have to pay attention to not step into one of the following three pitfalls: P6, inappropriate baseline; P7, inappropriate performance measures; or P8, the base rate fallacy.[10]

In the case of P6, the learning model is not compared against suitable baseline approaches. For instance, a simple, nonlearning-based method can sometimes achieve a similar or even better performance than a complex deep neural network. However, due to the lack of comparison, this fact remains hidden.

A similar problem arises if the chosen performance measures are not appropriate for the application scenario (P7). As an example, we often have to deal with highly imbalanced datasets in security, like in malware detection. In these cases, we have to identify malicious objects that represent only a small proportion of the entire data distribution. When using the wrong performance metrics in these settings, like the accuracy, one gets an entirely false estimate of the true performance of a learning-based system.

Moreover, even if proper metrics are used, the performance of a system might still be overestimated by ignoring the base rate of the negative class in reality (P8). Let us assume, for example, a seemingly efficient classifier with 99% true positives at 1% false positives. Yet, if we have a class ratio of 1:100, so that the negative class is predominant, even 1% false positives still cause 100 false positives for every 99 true positives.

### Deployment and Operation Phase

Finally, we obtain a learning model whose detection performance meets our requirements. We can now deploy and operate it in the wild. We might already assume that we have successfully avoided all possible pitfalls. Unfortunately, there are still two additional issues that can have a severe impact on the performance in practice.

First, we should account for any practical limitations that we did not consider throughout the evaluation. Oftentimes, new learning methods are solely evaluated in lab-only environments (P9), where crucial constraints of realistic settings are ignored, such as run-time or storage restrictions. As a result, a promising method might turn out to be unsuitable in a production setting. Furthermore, we need to consider the security of our learning-based system, as adversaries might run targeted attacks against it (P10). For instance, malicious actors could try to circumvent detection or derive information about the learning model. To fend off these attacks successfully, it is necessary to strengthen the learning model before its deployment.

## Are the Pitfalls Prevalent?

Naturally, the question arises of how likely each of the previously discussed pitfalls is to occur. To get an intuition, we review 30 academic papers published at top conferences for security between 2011 and 2020. When selecting the papers, we ensure that they cover a wide range of security-related topics, ranging from learning-based malware detection to intelligent vulnerability discovery. If a pitfall's presence is unclear, the reviewers decide conservatively and always give the authors of a paper the benefit of the doubt.

Figure 1 highlights the outcome of the study. The colored bar shows their prevalence in our study, with warmer colors depicting the presence of a pitfall. We find that the pitfalls are widespread even in top research. Each paper is affected by at least three of the discussed issues. The most prevalent pitfall is sampling bias (P1), followed by data snooping (P3), which are at least partly present in 90% and 73% of the considered publications, respectively. Similarly, other pitfalls occur frequently, such as the use of inappropriate performance measures (P7) or the evaluation in a lab-only setting (P9), both of which appear in at least 50% of all the papers. Interestingly, we find that that the presence of a pitfall is only accompanied by a discussion in 22% of the cases, indicating that there is a lack of awareness regarding these common issues.

To get a full picture of the situation, we have also collected feedback from the authors of the reviewed papers. The vast majority of the authors from which we received a response agreed that there is a lack of awareness for the identified pitfalls and confirm that these are widespread in security research.

## We Can Do Better

The discussed pitfalls are more than just an academic problem. In fact, they introduce severe biases and hinder actual progress in research. As a result, we need to discuss within the community how to overcome these problems in the future.

First and foremost, it is possible to avoid the identified pitfalls in many cases. Therefore, we recommend double-checking each stage of the machine learning pipeline and looking out for potential issues when developing a new approach. For instance, methods to fix inaccurate labels or methods of explainable AI to check for spurious correlations are applicable.

Unfortunately, there exist cases in which it can be challenging to avoid a pitfall entirely. As an example, it might be hard to compensate for sampling bias due to a lack of data. In these cases, it is crucial to openly discuss the problem so that other researchers can solve it in the future. In general, we thus recommend to "do your best" by mitigating pitfalls where possible and acknowledging remaining problems openly.

Overall, we hope that our work can help to promote sound research and bring the enormous potential of AI techniques into the reality of security. ■

## References


1. F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *Proc. USENIX Secur. Symp.*, 2019, pp. 729–746.
2. M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 263–274, doi: 10.1145/2660267.2660368.
3. S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in ML-based science," 2022, *arXiv:2207.07048*.
4. L. Cavallaro, J. Kinder, F. Pendlebury, and F. Pierazzi, "Are machine learning models for malware detection ready for prime time?" *IEEE Security Privacy*, vol. 21, no. 2, pp. 53–56, Mar./Apr. 2023, doi: 10.1109/MSEC.2023.3236543.
5. D. Arp et al., "Dos and don'ts of machine learning in computer security," in *Proc. USENIX Secur. Symp.*, 2022, p. 20.
6. Z. Li et al., "VulDeePecker: A deep learning-based system for vulnerability detection," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2018, pp. 1–15, doi: 10.14722/ndss.2018.23158.
7. Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 10,197–10,207.
8. S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep learning based vulnerability detection: Are we there yet?" *IEEE Trans. Softw. Eng.*, vol. 48, no. 9, pp. 3280–3296, Sep. 2022, doi: 10.1109/TSE.2021.3087402.
9. A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Evaluating explanation methods for deep learning in security," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, 2020, pp. 158–174, doi: 10.1109/EuroSP48549.2020.00018.
10. S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Trans. Inf. Syst. Security*, vol. 3, no. 3, pp. 186–205, Aug. 2000, doi: 10.1145/357830.357849.


**Daniel Arp** is a postdoctoral researcher in the Machine Learning Department at Technische Universität Berlin, 10623 Berlin, Germany, and a visiting researcher in the Systems Security Research Lab at University College London, WC1E 6BT London, U.K. His research focuses on machine learning for security and privacy. Arp received a Ph.D. in computer science from Technische Universität Braunschweig. Contact him at d.arp@tu-berlin.de.

**Erwin Quiring** is a postdoctoral researcher in the International Computer Science Institute at Berkeley, CA 94704 USA, and at the Ruhr University Bochum, 44780 Bochum, Germany. His research focus is to foster the reliability of machine learning in adversarial and nonadversarial settings, and his research includes studies of the application of machine learning to computer security, such as malware and fraud detection. Quiring received a Ph.D. in computer science from Technische Universität Braunschweig. Contact him at quiring@icsi.berkeley.edu.

**Feargus Pendlebury** is a visiting scholar at University College London, WC1E 6BT London, U.K. His research interests include developing machine learning techniques for hostile environments where adversaries seek to evade detection. Pendlebury received a Ph.D. in computer science from Royal Holloway, University of London, U.K. Contact him at fpendlebury@gmail.com.

**Alexander Warnecke** is a Ph.D. student in the Chair of Machine Learning and Security at Technische Universität Berlin, 10623 Berlin, Germany. His research interests include machine learning techniques for computer security applications, and focus on how to explain such systems and enrich them with uncertainty measures. Contact him at a.warnecke@tu-berlin.de.

**Fabio Pierazzi** is a lecturer (assistant professor) of computer science and the deputy head of the Cybersecurity group, Department of Informatics, King's College London, WC2R 2LS London, U.K. His research interests include systems security and machine learning, with a particular emphasis on settings in which attackers adapt quickly to new defenses. Pierazzi received a Ph.D. from the University of Modena, Italy. He is a Member of IEEE. Contact him at fabio.pierazzi@kcl.ac.uk.

**Christian Wressnegger** is an assistant professor of computer science at the Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany, heading the "Intelligent System Security" research group. Additionally, he is the speaker of the "KIT Graduate School Cyber Security" and PI at the "KASTEL Security Research Labs." His research focuses on combining the fields of machine learning and computer security. Wressnegger received a Ph.D. from Technische Universität Braunschweig and has graduated from Graz University of Technology, where he majored in computer science. Contact him at c.wressnegger@kit.edu.

**Lorenzo Cavallaro** is a full professor of computer science at University College London, WC1E 6BT London, U.K, where he leads the Systems Security Research Lab. His research focus is to enhance the effectiveness of machine learning for systems security in adversarial settings. Cavallaro received a Ph.D. in computer science from the University of Milan, Italy. Contact him at l.cavallaro@ucl.ac.uk.

**Konrad Rieck** is a full professor at Technische Universität Berlin, 10623 Berlin, Germany, where he leads the Chair of Machine Learning and Security. His research focuses on the intersection of machine learning and computer security. Rieck received a Ph.D. in computer science from Technische Universität Berlin. Contact him at rieck@tu-berlin.de.



*Digital Object Identifier 10.1109/MSEC.2023.3306197*