Challenges and Pitfalls in Malware Research

Marcus Botacin[1] Fabricio Ceschin[1] Ruimin Sun[2] Daniela Oliveira[2]
André Grégio[1]

[1]Federal University of Paraná (UFPR) – {mfbotacin,fjoceschin,gregio}@inf.ufpr.br
[2]University of Florida – gracesrm@ufl.edu – daniela@ece.ufl.edu

Abstract

As the malware research field became more established over the last two decades, new research questions arose, such as how to make malware research reproducible, how to bring scientific rigor to attack papers, or what is an appropriate malware dataset for relevant experimental results. The challenges these questions pose also brings pitfalls that affect the multiple malware research stakeholders. To help answering those questions and to highlight potential research pitfalls to be avoided, in this paper, we present a systematic literature review of 491 papers on malware research published in major security conferences between 2000 and 2018. We identified the most common pitfalls present in past literature and propose a method for assessing current (and future) malware research. Our goal is towards integrating science and engineering best practices to develop further, improved research by learning from issues present in the published body of work. As far as we know, this is the largest literature review of its kind and the first to summarize research pitfalls in a research methodology that avoids them. In total, we discovered 20 pitfalls that limit current research impact and reproducibility. The identified pitfalls range from (i) the lack of a proper threat model, that complicates paper's evaluation, to (ii) the use of closed-source solutions and private datasets, that limit reproducibility. We also report yet-to-be-overcome challenges that are inherent to the malware nature, such as non-deterministic analysis results. Based on our findings, we propose a set of actions to be taken by the malware research and development community for future work: (i) Consolidation of malware research as a field constituted of diverse research approaches (e.g., engineering solutions, offensive research, landscapes/observational studies, and network traffic/system traces analysis); (ii) design of engineering solutions with clearer, direct assumptions (e.g., positioning solutions as proofs-of-concept vs. deployable); (iii) design of experiments that reflects (and emphasizes) the target scenario for the proposed solution (e.g., corporation, user, country-wide); (iv) clearer exposition and discussion of limitations of used technologies and exercised norms/standards for research (e.g., the use of closed-source antiviruses as ground-truth).

1 Introduction

As the malware research field has grown and became more established over the last two decades, many research questions have arisen about challenges not yet completely overcome by the malware research community. For example, *How to make malware research reproducible?*; *Does papers based on attacks strictly require scientific rigor*?; *What constitutes a well-balanced and appropriate dataset of malware and goodware to evaluate detection solutions?*; *How to define a relevant, appropriated ground-truth for malware experiments?*; and *What criteria should be used to evaluate offline and online defense solutions?* are important questions for current and future research that cannot be left unanswered in further papers on the field. Failing in answering those questions may lead to pitfalls that cause delays in anti-malware advances, such as: (i) offensive research are not scientific enough to be published; (ii) the development of offline and online engineering solutions being developed and evaluated under the same assumptions; (iii) the adoption of unrealistic and biased datasets; (iv) considering any AV labeled samples as ground-truth without measuring their distribution and relevance, among others.

As pointed out by Herley and P. C. van Oorschot [82] in their survey of the security research field: *"The science seems under-developed in reporting experimental results, and consequently in the ability to use them. The research community does not seem to have developed a generally accepted way of reporting empirical studies so that people could reproduce the work"*. In this blurry scenario, all stakeholders are affected: beginners are often not aware of the challenges that they will face while developing their research projects, being discouraged after the first unexpected obstacles; researchers facing those challenges may not be completely aware that they are falling for a pitfall; industry experts often do not understand the role of academic research in security solutions development; paper reviewers end up with no guidelines about which project decisions are acceptable regarding the faced challenges, making it difficult to decide what type of work is good work.

Although understanding malware research challenges

and pitfalls is crucial for the advancement of the field and the development of the next generation of sound anti-malware security solutions, a few work have focused on attempting to answer those questions and shedding some light on these pitfalls. Previous works have only considered isolated malware research aspects, such as information flow [34] or sandbox limitations [127]. Therefore, in this paper, we decided to investigate such phenomena scientifically: we conducted a systematic review of the malware literature over a period of 18 years (2000-2018), which encompasses 491 papers from the major security conferences. Based on this systematization, we discuss practices that we deemed scientific, thus reproducible, and that should be included in the gold standard of malware research.

Overall, malware research integrates science and engineering aspects. Therefore, we describe "malware research" in terms of a malware research method, according to the following steps (see Figure 2): (i) Common Core (from the Scientific Method); (ii) Research Objective Definition; (ii) Background Research; (iii) Hypothesis/Research Requirements; (iv) Experiment Design; (v) Test of Hypothesis/Evaluation of Solution; and (vi) Analysis of Results. Based on this framework, we reviewed the selected literature and identified 20 fallacies about malware research, which were described and organized according to their occurrence in the Malware Research Method. The identified pitfalls range from the: (Reasoning Phase) unbalanced development of research work, which is currently concentrated on engineering solution proposals, with a scarcity of research on threat panoramas, which should provide the basis for supporting work on engineering solutions; (Development Phase) the lack of proper threat model definitions for many solutions, which makes their positioning and evaluation harder; and (Evaluation Phase) the use of private and/or non-representative datasets, that do not streamline reproducibility or their application in real scenarios.

In addition to pitfalls, we also identified challenges that researchers face regarding practical considerations, a step we modeled in our proposed Malware Research Method. For example, when developing a malware detection solution, researchers soon realize that many stakeholders, such as AV companies, do not disclose full information about their detection methods due to intellectual property issues, which limits solution's comparisons. Moreover, researchers working on dynamic analysis or detection also soon realize that a non-negligible percentage of their collected samples may be broken due to the lack of a required module, library, collaborating process, or because a domain was sinkholed, which also limits their evaluations. Therefore, we present a discussion about the root of each identified challenge and pitfall, supported by statistics from the literature review. Despite this strong supporting statistics, we do not consider all presented claims as definitive answers, but we acknowledge that others may have different understandings. Thus, we intend to stimulate the security community to discuss each pitfall and how they should be approached.

From the lessons learned, we proposed a set of recommendations for different stakeholders (researchers, reviewers, conference/workshop organizers, and industry), aiming at the next generation of research in the field and to discuss open challenges that the community has yet to tackle. For example, we propose **for researchers**: clearly state their assumptions about malware and goodware repositories to allow for bias identification and research reproducibility; **for reviewers**: be mindful of the Anchor bias [64] when evaluating the appropriateness of a dataset, since the representativeness of the environment in which an engineering tool is supposed to operate (corporate, home, lab) might be the most important evaluation criteria, despite the dataset size; **for conferences/workshop organizers**: increase support for the publication of more research on threat landscape, as they establish a foundation for new defense tools; and **for AV companies**: disclose the detection engines and/or methods leveraged for detecting samples as part of the AV labels. We do not expect that all of our proposed guidelines be fully addressed in all papers, since it can be almost impossible in some circumstances due to research/experiment limitations. However, our goal is to position them as a statement of good research practices to be pursued.

To the best of our knowledge, this is the first comprehensive systematization of pitfalls and challenges in almost two decades of malware research, and in which the pitfalls and challenges have been placed in the context of different phases of a proposed Malware Research Method, with a concrete actionable roadmap for the field. We limited our critical evaluation to the malware field, because it is the field we have experience as authors, reviewers and PC members. However, we believe that many of the points here discussed might be extrapolated for other security domains (along with the proper reasoning and adaptations), also providing a certain level of contribution to their scientific enhancement.

In summary, the contributions of our paper are threefold:

1. We propose a Malware Research method that integrates the scientific and engineering methods, which also addresses practical challenges of current technologies and industry practices.

2. We identify and discuss 20 Pitfalls in malware research, based on a systematization of 18 years (2000-2018) of malware literature (491 papers), with each pitfall placed in the phase they occur in the method.

3. We present a set of actionable recommendations for the field, researchers, reviewers, conference organizers, and industry, based on the lessons learned during the systematization performed.

This paper is organized as follows: Section 2 describes the methodology used in the literature systematization; Section 3 introduces the Malware Research method, a method for malware research integrating both scientific and engineering methods. Section 4 discusses 20 pitfalls in malware research contextualized according the phase they occur in the Malware Research method; Section 5 proposes actionable recommendations, based on learned lessons during systematization. Section 6 reviews related work and Section 7 concludes the paper.

2 Methodology

We systematized the knowledge of malware research to identify challenges and pitfalls during research development, whereas conducting research in the field. To avoid reporting anecdotal evidences, we support our discussion points with statistical data from the academic literature. To do so, we relied on PRISMA [159], an evidence-based minimum set of items for reporting in systematic reviews and meta-analyses, commonly used in the social sciences (see steps in Figure 1). Our goal is not to present a comprehensive survey of malware literature, which keeps growing at a fast pace [11, 163], but to systematize the most relevant pitfalls of the field in a scientific and reproducible fashion. Our search encompassed the most reputable repositories of peer-reviewed security papers (IEEE, ACM, USENIX, and Springer) published between 2000 and 2018.

There are multiple definitions of malicious behaviors [79, 116] that can be considered when performing malware research. In this work, we adopted the malware definition proposed by Skoudis and Zeltser [174]: "*Malware is a set of instructions that run on your computer and make your system do something that an attacker want it to do*", thus covering a broad range of threats, from self-contained apps to exploits. Therefore, the search focused on scholarly work indicating the keywords "`malware`" or "`malicious software`" in their titles, abstracts, or keywords. As each query resulted in a large number of papers (approximately 4,700 for IEEE, 2600 for ACM, 100 for USENIX, and 3,000 for Springer), we defined additional filtering criteria, such as paper's number of citations, and conference, workshop or journal ranking. As expected, papers in highly ranked conferences are significantly more cited than papers in other conferences or journal papers. Therefore, we selected the most cited papers in the set of top-ranked conferences.

We filtered out papers whose techniques could be employed by malware, but the contributions would not be mainly characterized as malware research, such as side-channel data exfiltration, hardware Trojans, and formal theorem proofs. In Table 1, we provide the distribution of the 491 selected papers by year and conference, highlighting the increasing pace of published papers over the years.

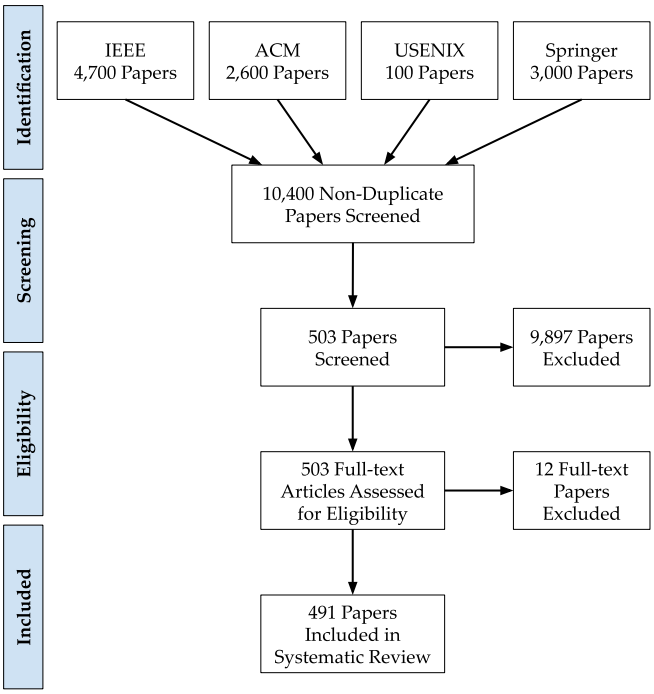

Figure 1: **PRISMA methodology.** Literature review steps.

The long-term observation period allows to spot trends and evolution in the malware research procedures. In this sense, it is important to highlight that our goal is not to point fingers to past research practices but to identify when they changed and if such changes produced positive results.

3 The Malware Research Method

In this work, we discuss malware research challenges and pitfalls based on the definitions of scientific methodology [157] and engineering design [149]. These principles introduce large commonalities and aim to evaluate the validity of a hypothesis or a solution to a given problem. However, the strategies and criteria for both methods differ in some aspects, for instance, there is no consensus on whether malware research falls into the science or engineering category. We propose that such characterization depends on the type of research, but in most cases, it is science **and** engineering. For example, the main goal of observational studies is science and not engineering, since there is no solution proposal for a problem. A framework to detect malware, on the other hand, aims at both science and engineering.

Currently, there is no consensus on a methodology for conducting malware experiments, with some work evaluated exclusively according to one of the aforementioned methods. This view can lead to many derived pitfalls, as computer science presents many paradigms [52] which cannot be evaluated in a complete manner using a sin-

Table 1: **Selected Papers**. Distribution per year (2000 – 2018) and per venue.

Venue/Year	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total
USENIX (Security, LEET & WOOT)	1	0	0	0	0	1	1	6	2	3	7	8	10	12	9	7	9	13	6	95
CCS	0	0	0	0	0	0	0	2	4	6	6	7	11	9	11	14	2	11	6	89
ACSAC	0	0	0	0	2	3	2	4	4	1	3	8	10	7	10	6	3	7	8	78
IEEE S&P	0	1	0	0	0	1	3	2	1	0	0	10	17	12	3	6	4	5	3	68
DIMVA	0	0	0	0	0	4	4	3	8	2	3	0	8	4	8	7	7	5	4	67
NDSS	0	0	0	0	1	0	2	0	3	3	3	3	2	4	5	4	9	7	3	49
RAID	0	0	1	0	0	1	3	0	0	0	0	0	3	5	5	3	4	3	3	31
ESORICS	0	0	0	0	0	1	0	0	2	1	0	0	2	3	3	0	1	1	0	14
Total	1	1	1	0	3	11	15	17	24	16	22	36	63	56	54	47	39	52	33	491

gle metric (e.g., a validated engineer prototype that does not ensure scientific proof for all scenarios, or a scientifically proved theorem that cannot be implemented as a prototype).

As mentioned in the Introduction, we propose that malware research should follow an approach as effective as NASA's, i.e., one that combines and benefits from both methods (scientific and engineering). Whereas NASA's ultimate goal is to promote research exploration by scientific means [143], most of its efforts (and budget) are spent on developing technology to overcome the practical challenges involved in such discoveries [144]. Therefore, we here propose that malware research should be conducted integrating the scientific and engineering methods in what we call the **Malware Research Method**.

In Figure 2, we illustrate this integrated proposal, detailing the typical research workflow for developing malware research, which is composed of the following steps:

1. **Common Core:** This step is common to both scientific and engineering methods, and consists of defining what type of research will be conducted and how the research will proceed.
   (a) **Research Objective Definition (*What do you want to do?*):** This step consists of establishing the goals of the proposed research. For example, does the research consist of proposing a new anti-malware solution, a new attack, or a new tool to aid malware analysis?
   (b) **Background Research (*What have already been done in previous work?*):** This step consists of gathering background information to support hypothesis formulation and solutions requirement identification. It also allows the research to be placed in the context of prior literature to assert the type of its contribution, e.g. novel work, incremental research, reproducibility work, etc.
   (c) **Hypothesis Definition & Research Requirements (*What are the proposed hypothesis or requirement steps to test them?*):** Depending on the type of research, this step consists of formulating hypotheses to be tested and/or defining the requirements to test the hypothesis, which may include developing an engineering solution.
2. **Engineering:** This step is required only for research that proposes practical/empirical solutions to problems, whose results do not fit into the classic scientific method [157]. However, we chose to include it to cover the bulk of malware research that consists of the development of tools that, in the end, support further scientific advances. Non-engineering research (e.g., observational studies, analysis of samples) will skip these steps and proceed directly to "Experiment Design".
   (a) **Solution Requirements:** This step consists of reasoning and stating the functional, security, operational, performance, and/or usability requirements of the proposed solution.
   (b) **Solution Design (*What are the solution requirements and how to implement them?*):** This step consists of reasoning about design aspects of the proposed solution, such as the definition of a threat model, assumptions, target platform (e.g., Linux, Windows, Android, IoT) and public (e.g., corporate, home users, mobile users, researchers, etc.), and definition of whether the solution is a proof-of-concept prototype or is proper for deployment into production.
   (c) **Solution Development/Prototyping (*How to develop the proposed solution?*):** This step consists of effectively implementing the proposed solution.
3. **Scientific Method:** This step consists of testing hypotheses and analyzing the results obtained (for non-engineering malware research) or performing an empirical evaluation of the proposed solution.
   (a) **Experiment Design (*How will you evaluate what you did?*):** This step consists of designing an experiment to test hypotheses or to verify whether the established requirements of the proposed solution were met or not. For purely scientific studies, it involves determining the methodology for data collection

or performing an attack, sample sizes, experiment conditions, and dependent and independent variables. If the study involves humans, the researcher also needs either to obtain institutional review board (IRB) approval for data collection or to justify the reason behind the lack of need to seek such approval. For engineering solutions, it involves determining evaluation criteria (e.g., functionality, security, performance, usability, accuracy) and defining the test environment, benchmarks, and datasets.

(b) **Tests of Hypothesis/Evaluation of Solution (*What happens in practice?*):** This step consists of testing hypotheses or effectively conducting experiments to evaluate an engineering solution by leveraging the developed or existing tools and considering the peculiarities, limitations, and idiosyncrasies of supporting tools, technologies, environments, operating systems, and software repositories. Depending on the type of research, it may leverage statistical methods to test hypotheses, or the use of AV solutions, benchmarks, and virtual machines/sandboxes. Practical considerations of evaluation procedures are often neglected in most research works. In addition, whereas we highlight the importance of considering practical aspects for performing experiments, we advocate for practical considerations to be considered in all research steps.

Once we have discussed our method for malware research, it is important to highlight that such dichotomy between science and engineering has not been restricted to malware analysis, but, in fact, extends to the whole computer science field [58]. However, in this work, we limit the discussion to the malware subject since our literature review is limited to it.

In the following section, we discuss the major challenges and pitfalls of malware research, which we identified after applying the aforementioned steps, and based on our extensive review of the last (almost) two decades of literature in the field, as well as during our practice developing malware analysis and detection malware research experiments.

4 Challenges & Pitfalls

This section presents the challenges and pitfalls of malware research according to our proposed Malware Research Method and the literature review. Each pitfall is discussed as a self-contained issue, although some of them might also relate to others. An overview of the discussed challenges and pitfalls is depicted in Figure 3.

4.1 Research Objective Definition

"Malware research" is a term used in the literature to describe a wide field of work that embraces multiple goals. Therefore, before digging into any detail about how research is conducted, we need to understand which types of research are developed under the malware "umbrella". To provide a summary of the paper's goals, our research team read and (manually) classified all papers. According to our understanding (cross-checked by the research team's members), malware research can be categorized as follows (see Table 2 for examples of representative work in each category):

1. **Engineering Solutions:** Research proposing mechanisms to counter malware, such as signature-based and behavioral-based detectors and tools and frameworks to aid malware analysis (e.g., static analyzers, sandboxes, reverse engineering frameworks).

2. **Offensive Techniques**: Research exposing gaps and/or security breaches to inform the development of future effective security solutions. It involves presenting exploits and proofs-of-concept (PoCs) attacks against specific targets.

3. **Observational Studies**: Research focusing on analyzing samples from a family of malware, specific platform (e.g., desktop, mobile), or ecosystem landscape (e.g., a country, corporation), to inform the development of future security solutions developments tools and malware trends. This type of research usually falls exclusively under the scientific method, therefore, skipping the steps from the engineering method.

4. **Network Traffic**: Research analyzing and proposing solutions for detecting malicious payloads in network traffic. It might or not involve the execution of malware samples, because malicious traffic can be obtained from multiple sources (e.g., honeypots, corporation, etc). Despite not directly investigating malware samples, such work advances understanding in malware detection via network behavior characterization. Most network traffic research work skips typical malware analysis research issues, such as system interaction, packing, and anti-analysis, to focus on the traffic generated by successfully executed samples. Therefore, we do not consider this type of study for system statistics pitfalls to not bias our analyses (e.g., with their lack of malware threat models), but we considered them as malware research for the sake of dataset size definition evaluation.

These categories are not an exhaustive enumeration of all types of malware research, but a summary of the most popular research types. First, because some types of

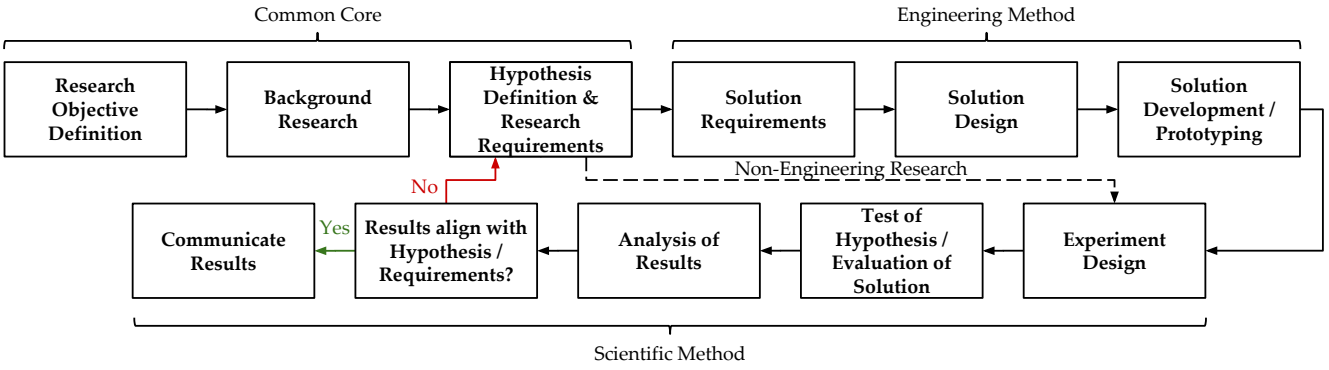

Figure 2: **The Malware Research Method.** Integration of the scientific and engineering methods.

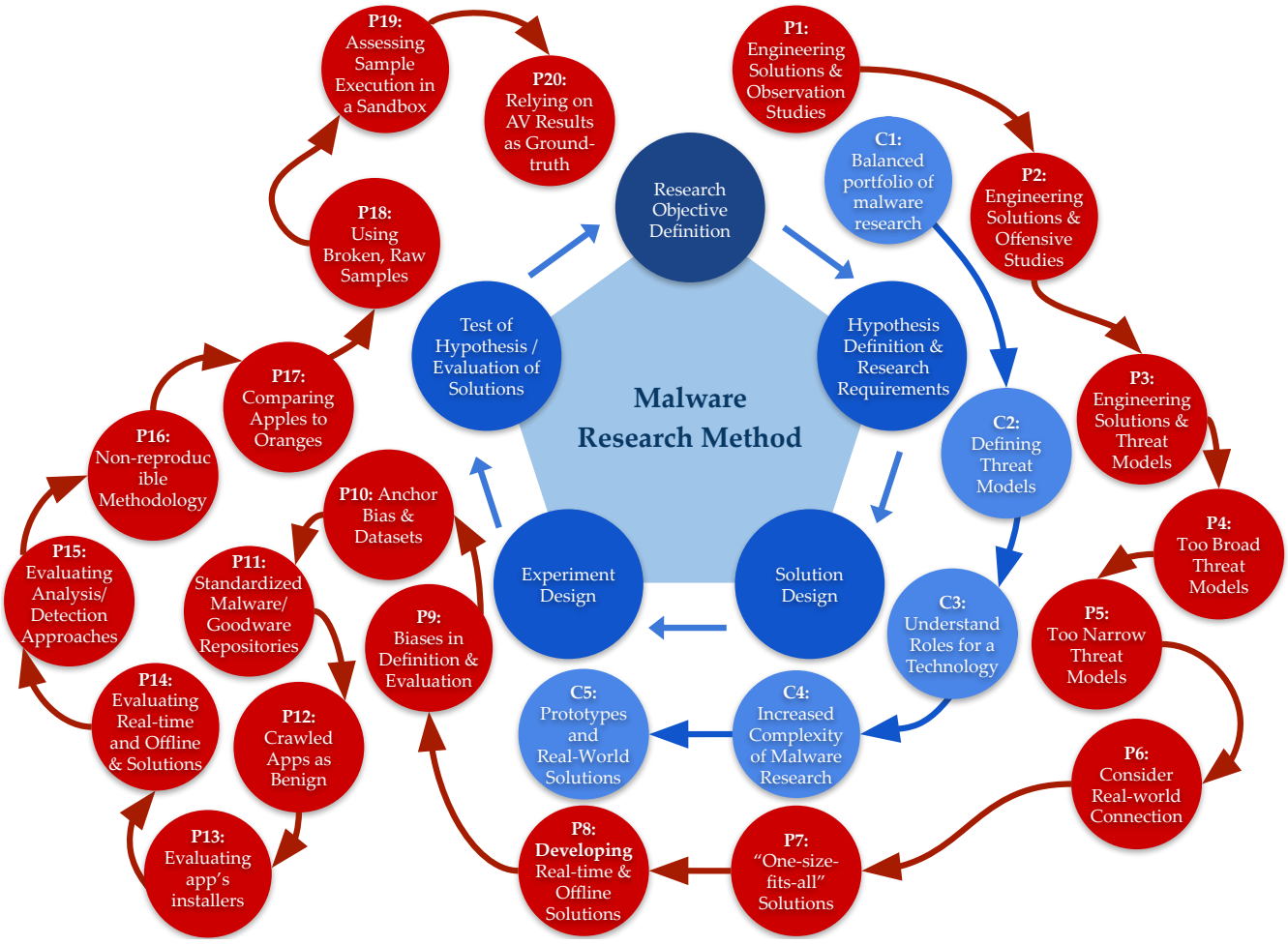

Figure 3: **Overall Paper Organization.** Challenges (blue) and Pitfalls (red).

Table 2: Representative papers for each research type.

Objective	Representative Papers
Observational Studies	Desktop Malware [13], [14, 54, 147, 125, 113, 29, 187] Mobile Malware [126], [213, 74, 59, 211, 62, 100] Web Malware [165], [4, 138, 142, 140]
Engineering Solutions	Sandbox [104] [203, 19, 137, 21, 26, 209, 92, 136, 75] Malware Detector [150], [44, 47, 180, 202, 84, 65, 208, 114] Family Classifier [86], [57, 212, 151, 107, 155, 77, 97, 206, 106, 181]
Offensive Research	Targeting System Design: [71], [32, 198, 55, 108, 12, 28, 204, 93, 6] Targeting Firmware: [27], [158, 117, 81, 40, 15, 175] Adversarial Machine Learning [42], [78, 115, 207, 91, 45, 43, 210, 102, 2, 96]
Network Traffic	Domain Generation Algorithms [182], [195, 148, 134, 17, 83, 169, 197, 120] Honeypots [10], [119, 94, 214, 177, 88, 205, 118, 51, 70, 76, 89] Network Signatures [161], [200, 156, 162, 141, 160, 146, 123]

research might not be represented by the papers published in the conferences during the considered period (e.g., malware propagation models). Second, because, in practice, many research work outside the system security field exemplifies their solutions via PoCs dubbed as malware. However, their main contributions are placed outside the malware domain, therefore we do not classify them here as malware research. For instance, we did not include work on cryptographic side-channels or attacks to air-gap systems as malware research, because the primary goal of these proposals are not to present new ways of infecting systems but to discuss information theory-based techniques for data retrieval.

It is also important to notice that these categories are technology-agnostic, i.e., their goals might be accomplished using distinct techniques. For instance, machine learning and deep learning techniques are often associated with malware detection tasks, but they can also be considered for traffic analysis or even attack purposes.

Challenge 1: Developing a balanced portfolio of types of malware research. Considering the Malware Research method, it is plausible to hypothesize that observational studies would be the most popular type of malware research, given that understanding malware (characteristics, behavior, invariants, targets, trends) should precede the development of solutions. Insights from such studies can inform the impact of vulnerabilities, the evaluation of defense solutions, and the understanding of context and real-world scenarios. Similarly, one would expect a greater number of offensive research to be proposed because such type of research helps in anticipation of threats, identification of gaps in observational studies, development of sound defense solutions for novel threats. However, from 2000 to 2004, **only** engineering solutions were published in the literature. Figure 4 presents research type distribution after 2005.

Engineering Solutions have been the most popular

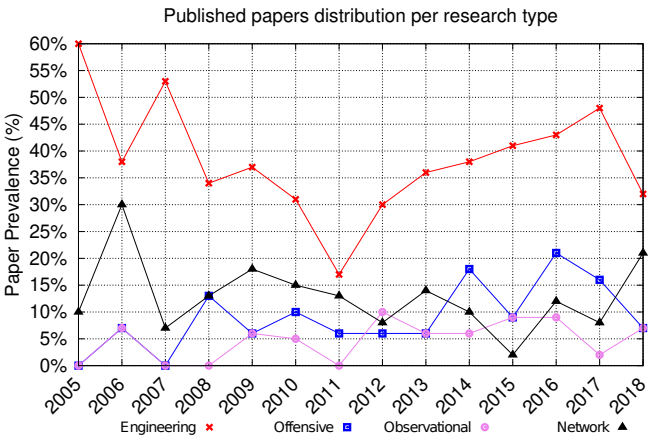

Figure 4: **Prevalence of published paper as a function of research type.** For most years, Engineering Solution has been the most prevalent research type, whereas Observational Studies has been the less popular.

research type over the years, thus indicating the consolidation of this research type. Network Traffic research is the second most popular research type in almost all years, thus also suggesting its consolidations as a long-term research line. Observational Studies and Offensive Research, in turn, have been the least popular, although having recently (2014 and 2016) started to grow, suggesting that this kind of research is not consolidated yet as a long-term research line. On one hand, it is desirable that the community prioritizes fighting malware with concrete solutions to the multiple threats targeting user's security. On the other hand, this significant disparity between the prevalence of papers on Engineering Solutions vs. the remaining types (e.g., observational and offensive research work combined account for less than 50% of all published Engineering Solutions papers) raises concerns about the effectiveness

of such approaches—they might address less relevant aspects, factors, and problems due to a lack of insights about real-world malware trends. How can one be sure about the effectiveness of proposed solutions without long-term evaluations of their impact? Therefore, it is essential for the community to diversify and understand the impact of the distinct types of malware research work, as following discussed.

Pitfall 1: Engineering Solutions Not Informed by Observation Studies Data. The current paucity of observational studies in the academic literature led us to investigate whether this comes (i) from few existent observational studies being enough to inform multiple engineering solutions or (ii) engineering solutions being developed without considering observational studies findings, which would consist in a pitfall since engineering solutions developed without a good understanding of context and prevalence of families may not reflect real-world scenarios. Further, this might challenge the definition of appropriate dataset sizes (e.g., how many different samples, on average, target a user within a given period and in a given environment?), malware family[1] balancing (e.g., are corporations more targeted by Trojans or Ransomware?), and threat model definition (e.g., what is the prevalence of kernel rootkits vs userland malware?). Lack of reliance on observational studies as foundations for developing engineering solutions may also cause paper reviewers to acquire biases. For example, if the study does not leverage the development of a practical solution, some reviewers might claim that the contribution is limited.

Although our literature review revealed the existence of observational studies (e.g., malware packer [187], Windows malware [13], and Android malware [126]) that could be used to back many project decisions (e.g., based, for instance, on the threat prevalence data presented by these research work), their use is very limited in practice. Whereas each one of these papers is cited by more than 10 other papers among the considered top conferences, their citations are placed in the context of related work for many engineering solutions [80, 46] and not on project decision's support.

In our view, a good usage of previous observational studies is when their findings are used to support project decisions. Although no good example of this phenomenon was identified among the considered malware papers, we can identify this good practice in the study of Levesque and Fernandez [121], which presents an experiment to assess the effectiveness of an anti-malware solution for a population of 50 users via clinical trials. They describe their assessment steps as follows: They (i) first describe the dataset size definition challenge ("*The challenge is then to identify the desired effect size to be detected before conducting the experiment*"); (ii) identify that the challenge can be overcome by relying on previous data ("*the effect size can be estimated based on prior studies*"); and (iii) finally, leverage this prior data for the task at hand ("*Based on the results of our previous study...we know that 20% of the participants were infected even though they were protected, and that 38% of the total population would have been infected if they had not been protected by an AV product.*").

Whereas the lack of longitudinal studies was already acknowledged for some research subjects (e.g., Luo et al. [129] claiming "*there is no longitudinal study that investigates the evolution of mobile browser vulnerabilities*"), we here extend those claims for the general malware research subject.

Pitfall 2: Engineering Solutions Not Informed by Offensive Studies Data. As for the observational studies, the paucity in the number of offensive papers published in the academic literature led us to question whether (i) few studies were enough to support engineering solution's developments or (ii) solutions have been developed without being informed by such type of work. We discovered that, as for observational studies, offensive papers have been mostly referred as related work and not as a basis for developments.

A first hypothesis for the lack of reliance on offensive papers is a generalization of the reasons for the lack of reliance on observational studies, with researchers becoming used to develop a hypothesis in an ad-hoc manner. Another plausible hypothesis for the relatively low number of published and referred offensive work are research biases. Previous work have already discussed the existence of possible biases in favor and against offensive papers [82]; Some researchers consider that vulnerability discoveries (also called "attack papers") are not scientific contributions. On one hand, we agree that disclosing vulnerabilities without appropriate reasoning (and responsible reporting to stakeholders) does not contribute towards advancing the field. On the other hand, we believe that the field (especially defense solutions) can greatly benefit from more offensive work conducted in a scientific manner (i.e., constructing hypotheses and theories in addition to presenting an isolated evidence). Examples of open or only partially-addressed research questions for offensive papers are: research exposing weaknesses in existing defense solutions (e.g., malware classifiers evasions), insights on attacks' measurement in practice (e.g., how long do attackers take to exploit a vulnerability in practice after a 0-day disclosure? [16]) insights to inform the development of future defensive solutions (e.g., how hard is it to find a Return Oriented Programming–ROP–gadget in a program?).

The case of ROP attacks, in a general way, is an illustrative example of offensive papers developed in a scientific manner according the Malware Research Method proposed in this work. Whereas proposing exploitation

[1] set of samples with similar goals and/or implementations

techniques, these research work [132, 33, 71] do not focus on exploiting specific applications but to investigate a whole class of vulnerabilities abusing the same infection vectors (e.g., buffer overflows and code reuse). The work by Goktas et all [71], for instance, reproduces and investigates previous solutions proposed in the literature to establish the limits of existing ROP defenses.

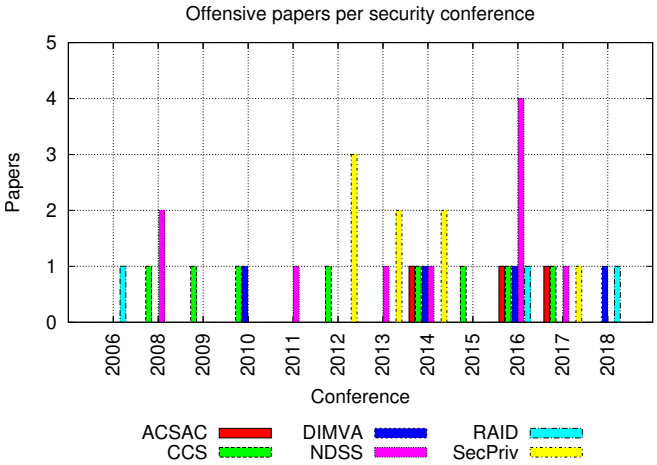

Figure 5: **Offensive papers per security conference.** Most malware research papers are published in USENIX WOOT and not in the other top venues.

Regardless of the reason for the low number of offensive malware papers, we advocate for the community to acknowledge the importance of this type of study and also focus its efforts on the development of more offensive research. Currently, a first step towards acknowledging and increasing the importance of the offensive security field has been given by the establishment of targeted events as USENIX Workshop On Offensive Technologies (WOOT). However, the community efforts should still be extended to other top venues, which do not present offensive malware papers published in most years, as shown in Figure 5.

4.2 Hypothesis Definition & Research Requirements

The lack of reliance on malware landscapes and other threat panoramas to support the proper design of malware research projects may end up in development pitfalls that require additional reasoning to be overcome, as following discussed:

Challenge 2: Defining Threat Models. Threat modeling defines: (i) what will be protected (e.g., a system against malware, a buffer against injection, etc.), (ii) which methods will be leveraged for the task (e.g., static analysis, runtime monitoring, machine learning classification) and (iii) who the stakeholders are (e.g., user, analyst, system administrator, an AV, company, attacker, etc).

The threat model should reflect the decisions about the question or problem at hand, for example, which problem should be addressed first and which the most promising strategies for testing the hypothesis or solving the problem.

A well-defined threat model allows researchers to better position their work in the context of the literature by clearly stating the question(s) that they want to answer or the problem that they are trying to solve and, therefore, streamlining the peer-review process evaluating whether the researcher's goals were achieved.

Therefore, proposed research without clearly defined threat models also makes the peer-review process harder and raises concerns about the viability and limits of the hypotheses and proposed solutions.

Threat model definitions should not be limited to papers proposing defensive solutions, but should also cover attack papers. In such a case, researchers are required to clearly define what are the assumptions about the attack (e.g., infection vector) and which type of defense the attack is supposed to bypass (e.g., address space layout randomization).

Pitfall 3: Engineering solutions and offensive research failing to define clear threat models. Ideally, all malware research papers should dedicate space for addressing threat model definitions and/or papers assumptions, either in the form of a dedicated threat model section or as any other portion of the text clearly highlighting researcher's reasoning on the subject. Unfortunately, this is not observed in practice. Figure 6 shows the ratio of engineering solutions and offensive research papers published after 2007 presenting threat model definitions (Model line). In our review, we were not able to identify any paper explicitly defining threat models in a structured way (e.g., a section or paragraph exclusively dedicated to the present researcher's reasoning) from 2000 to 2006 (which is likely due to the fact that this concept was not well-established by that time).

We notice that, in practice, most papers do not present a dedicated threat model section, either by distributing solution presentation along with the entire text in a non-structured manner or even by not reasoning about the proposed solution threat model, assuming some implicit model and/or standard, which is not always clear for the reader. For instance, in some papers [103, 193], the reader only discovers that Windows was the targeted OS when a Windows API is referred, which indicates an implicit assumption on the popularity of Windows malware over that period, an important missing information to motivate the work, evaluate their importance, and characterize their results.

This lack of formalization is understandable, however, when the field was establishing itself in the early 2000s, but today, with the relative maturity of the field, crucial that malware research follows a more rigorous scientific-

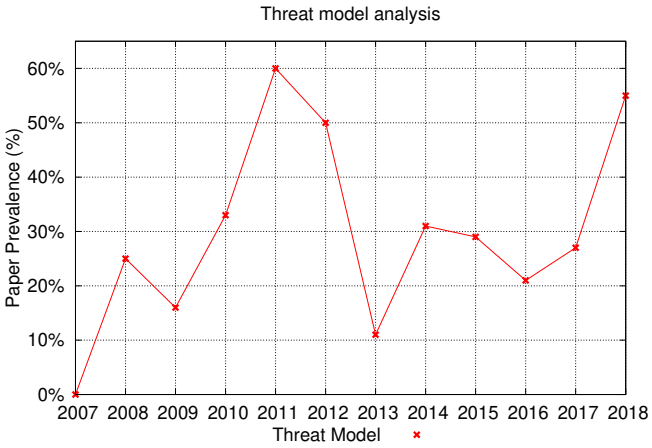

Figure 6: **Threat modelling.** The number of papers formalizing threat models have been growing, but it still corresponds to roughly 50% of the published work (Pitfall 3).

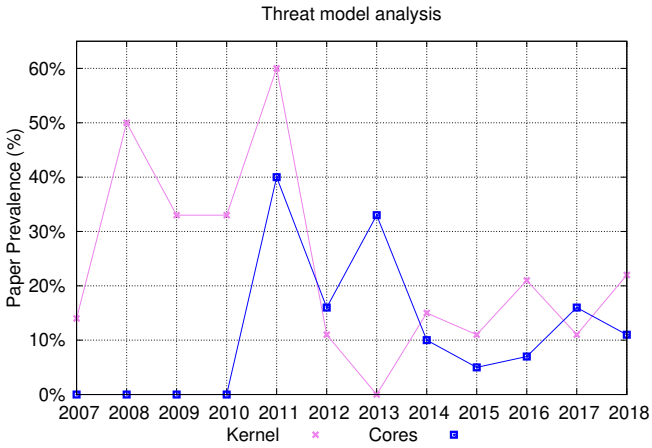

Figure 7: **Threat modelling.** The percentage of published work whose threat model explicitly states the addressing of kernel issues is very low, oscillating in the range below 20% in recent years (Pitfall 4). Also, the number of papers explicitly stating, in their threat models, whether their solutions is intended for single or multicore is low, less than 20% in recent years.

engineering method. Fortunately, such a trend has been observed, with an increase in papers including threat model sections in the last decade (2008 to 2018). in comparison to the scarcity of definitions from the beginning of the 2000s. We highlight that a significant fraction of papers lacking a threat model section are offensive papers, with ≈50% of attack papers not describing clearly what are the security mechanisms intended to be bypassed.

Although defining a threat model is essential, there is no "gold rule" for defining a precise threat model and the malware field did not adopt any particular approach (other security fields, such as cryptography, have some popular threat modelling strategies [61, 122]). Therefore, whereas making the correct decisions is hard, making mistakes is easy and might lead to security breaches, as following discussed.

Pitfall 4: Too broad threat models. Defining a threat model is challenging, therefore pitfalls might arise even when a threat model is clearly stated. For instance, researchers and reviewers might exaggerate when defining and evaluating the required security coverage of the proposed engineering solutions and/or attack proposals. For instance, an important aspect of threat model definition is determining what entity will be protected or attacked (e.g., userland vs kernel), i.e., what the solution scope is. Typically, current systems will either protect userland or kernel land. Therefore, researchers should explicitly state their choices about their solution's operational scopes. Figure 7 shows the prevalence of solutions and attack papers explicitly stating whether or not their proposal addresses kernel space (Kernel line), even if in an unstructured manner. From 2004 to 2010, it was more common than in recent years (2010 to 2018) for engineering papers to state Kernels somehow in their scope.

One hypothesis for this early popularity is the lack of consolidation of the practice of stating threat models in a more structured fashion, where issues regarding porting a solution to or conducting an attack at the kernel level are directly confronted. When threat models are not defined, authors address issues in a free,non-systematized way, only mentioning that a solution port to a given scenario was possible, but without discussing it in proper details. We were able to find many examples in the literature of this practice in multiple contexts: (i) "dynamic analyses could be easily ported to X-Force" [154]; (ii) "the techniques can be easily ported to support Linux" [56]; and (iii) "can be easily ported to handle PE binary on Windows" [124]. In practice, porting a solution presents multiple implications that should be discussed in details (see Section 4.4).

In turn, when threat models are clearly defined, particularities, such as the feasibility of implementing a kernel version of a given attack or solution, are omitted by definition when they are out of scope, thus making the researcher focus only on the proposed scenario, in a more rigorous manner.

In recent years, after threat models started being more clearly defined, the prevalence of userland solutions has increased, which is hypothesized to be a more realistic scenario since userland malware is easier to implement than kernel threats. Although such a hypothesis is plausible, it is hard to evaluate how close this trend reflects real scenarios because of paucity of data supporting the prevalence of userland threats in multiple scenarios, which affects not only researchers defining threat models, but also reviewers evaluating papers. For

example, a reviewer might be more prone to pinpoint as a weakness for a particular solution to not addressing kernel threats. However, why should it be necessary for a proposed userland solution to also address kernel space? The relevance of a threat model (e.g., addresses only userland) should be backed by data indicating the relevance of the proposed threat model. We are not claiming that privilege escalation is not a significant threat in some scenarios, but we consider that reviewers questioning the contribution of a solution because it does not address a variety of scenarios (e.g., userland and kernel land, desktop and mobile) might still be a bias derived from early years of poorly-defined threat models and the current paucity of observational studies providing insights about prevalence and relevance of threats.

Pitfall 5: Too narrow threat models. If on the one hand, researchers and reviewers might exaggerate the security coverage requirements for the developed solutions, on the other hand, they might neglect important aspects.

For instance, another important threat model definition is how a given scope will be protected. Modern architectures have been evolving over years from single-core processors to multi-core architectures. Therefore, it would be natural for both attackers and defenders to target this scenario. Many solutions, however, have still been developed for the old single-core paradigm [172]. Figure 7 shows the prevalence of papers stating whether their solutions are intended to operate on single or multi-processed platforms (Core line). Most work does not state their assumptions regarding the processor, which made us assume that such solutions do not address multi-core issues. Therefore, this (assumed) prevalence of single-core solutions shows that, in the core aspect, reviewers have not been challenging solutions to address broader threat models, as observed in the case of userland-kernel's case.

In practice, it reflects the lack of supporting data regarding the prevalence of threats in different architectures and the lack of evaluation of the real impact of multi-core threats and solutions in actual scenarios. Note that, we are not questioning the contribution of single-core-based solutions, which are valid PoCs (see Section 4.3), but actually pointing out that not properly defined threat model may lead to development gaps, such as the lack of incentives for the understanding of the impact of distributed threats and the development of multi-core-based security solutions, problems not completely addressed by previous work [130, 90, 23].

Challenge 3: Understanding the Roles for a Technology. Although the discussion on solutions implementations is placed in another step of our proposed malware research process, we believe that the adoption of a technology and/or approach is still part of the threat model discussion, because the drawbacks of a technology must be compatible with the scenario envisioned by the researchers and/or users. If these are handled separately, the greater the chances of solutions not fulfilling the requirements and of research pitfalls emerging. Thus, researchers must understand what is the role of each technology in the "big picture" of a security solution. It is essential to understand and acknowledge the pros and cons of each technology (e.g., signatures, heuristics, machine learning).

Considering the machine learning (ML) technique as an example, due to its recent popularity in the field (in conjunction with deep learning and other variations), researchers must have clear in mind that it might be applied in distinct steps of a security process and for distinct tasks, with each own of them presenting their own drawbacks. In the context of this work, ML technique is mostly (but not only) referred to as malware detection solutions, but the approaches for this vary significantly: from the static classification of files using feature vectors [36] to the dynamic monitoring of the whole-system operation using outlier detection algorithms and temporal series [99]. Therefore, each case presents its own drawbacks to be evaluated, such as the distinct limitations (e.g., packing in the first vs. performance in the latter), and/or distinct competing technologies (e.g., signature in the first vs. hardware counters in the latter).

Section 6 points to distinct surveys on the drawbacks of ML for security applications. In the following, we discuss the most common pitfall derived from the comparison of distinct technologies (including ML).

Pitfall 6: Failing to consider real-world connection. Ideally, academic research should introduce proposals that can be further adopted by industry and/or by home users. However, evaluating if a proposal is ready to transition to practice is hard. For example, consider the signature-matching malware detection paradigm. Whereas signature-based approaches are generally proven evadable by morphing code techniques [185], this paradigm is still widely used by AV companies [48, 50, 18], as the fastest approach to respond to new threats. Considering this scenario, should signature-based detection research still be considered in academia?[2]

We were not able to identify the ratio of defensive solutions leveraging behavior-based and signature-based approaches for the papers published before 2005, as these did not clearly state their detection methods. Figure 8 shows the prevalence of signature-based and behavioral-based defensive solutions leveraging behavior-based and signature-based approaches for the papers published after 2006. Most research work tackling malware detection leverage behavior-based techniques (60% of all papers proposing a malware detection solution and 70% in the last eight years) instead of signature-based ap-

[2] In our view, it should.

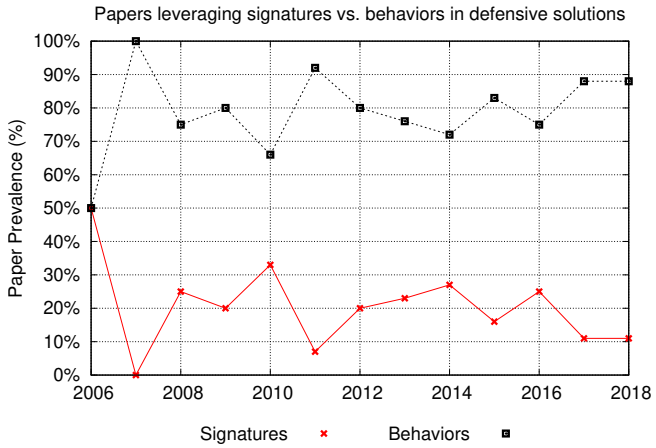

Figure 8: **Prevalence of papers proposing signature-based vs behavioral-based detection.** Behavior-based approaches are more prevalent than signature-based approaches.

proaches. Does this effort distribution reflect real-world needs? This is a hard question, as the current understanding about AV detection methods and statistics is limited (see Section 4.5 for a comprehensive discussion). There are many pros and cons that should be considered when evaluating the appropriateness of signatures and other behavior mechanisms (e.g., performance, database size, so on). Table 3 exemplifies the comparison of some selected research work to highlight their multi-dimensional characteristics. However, despite the distinct pros and cons of signatures and ML models, the most frequent complaint about signatures seems to be their low resistance to evasion attempts by minimal morphing code patches (i.e., changing a few bits). This is mentioned in almost all work proposing behavior-based solutions. Whereas this is indeed a limitation of signature-based approaches, this kind of attack is also possible for other approaches, such as the ones based on Machine Learning (ML), which is not mentioned by any of the evaluated work. A recent research work demonstrates that the simple fact of appending bytes to a binary might lead to classification evasion [36] (for a more complete discussion on ML attacks, check [186]). Thus, this evasion possibility should not be the only criteria to consider or discard signature or ML approaches as detection mechanisms.

In practice, whereas some claim that "*signatures are dead*" [170], some AV companies incorporate YARA signatures as part of their solutions [133]. From our literature review, the academic community seems to be more engaged in the first hypothesis, given the prevalence of behavior-based research work. The community, however, should care to not neglect the second scenario and acquire a bias against new signature-based proposals. In real scenarios, signatures and behavioral approaches tend to be used complementary, and research work targeting real-scenarios must reflect this setting.

4.3 Solutions Design

The research goals and considerations defined in the previous steps directly affect the defensive solutions developed to achieve them. Here, we discuss the pitfalls derived from vague/imprecise research goals and design assumptions in defense-based engineering tools.

Challenge 4: Handling the Increased Complexity of Malware Research. Every research discovery offers contributions to the security community, either by introducing a new technique, proving a security property, or providing data about a given particular scenario. Therefore, with the maturity of the malware research field, it is plausible to hypothesize that research work has been increasingly complex and proposing a greater number of contributions.

To explore this hypothesis, we *manually* aggregated the number of claimed contributions from all 491 papers reviewed as part of this systematization. Figure 9 shows that the average number of claimed contributions per paper per year has been increasing over time and seems to have saturated in an average of three distinct contributions per paper since 2014.

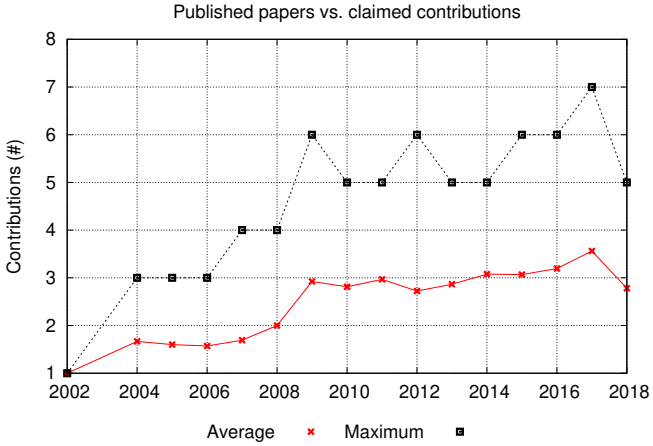

Figure 9: **Number of papers claimed contributions**. Papers are getting more complex and claiming an increasing number of contributions.

On one hand, a growing number of contributions per paper is a good indicator that the field has been tackling significant challenges and addressing bigger problems. On the other hand, this increasing number of claimed contributions per paper raises the following question: *are solutions claiming multiple and diverse contributions attempting to operate in a "one-size-fits-all" fashion"?*

It is important to notice that we are not doubting these researcher's capabilities, but it seems that presenting such a high number, such as six or seven, of multiple, distinct contributions in a single paper is not reasonable

Table 3: **Research Works Comparison.** Research works relying on distinct approach must be evaluated according to their multiple dimensions.

Work	Gionta et al [69]	Cha et al. [39]	Shafiq et al. [173]	Allen et al. [3]
Goal	Triage	Triage	Detection	Detection
Technique	Signature	Signature	Model	Model
Environment	Cloud	Linux	Windows	Android
Features	✗	✗	✓	✓
Performance (+)	87%	90%	0%	0%
Detection (+)	0%	0%	96%	97%

when aiming to provide a complete scientific treatment of the investigated subject. We are also not suggesting this metric to be used as definitive proof of the quality of one's work; it is not possible, distinct authors have distinct writing styles when stating their contributions. More specifically, we are concerned about the hypothetical possibility of this "raising the bar" mentality on claimed contributions creating a scenario of discoveries being less in-depth explored than they should. It is understandable that in the current very competitive world researchers have to tune up their claims, but it cannot be done at the charge of the discovery and exploration feelings. Thus, authors should care to first explore their discoveries in-depth and demonstrate their potential for solving the tackled problem in the stated scenario, despite limitations to operate in other conditions, before attempting to extend their solutions to other contexts.

Pitfall 7: Developing "one-size-fits-all" solutions. To understand the problem regarding "one-size-fits-all" solutions, consider an AV solution advertised as having multiple operation modes: (i) a static-signature matcher, which is fast, but vulnerable to morphing code samples; (ii) a runtime, behavior-based monitor, which is effective, but imposes significant performance overhead; and (iii) a cloud-based agent, which is effective, presents low overhead but incurs significant detection delays because of its need to upload suspicious objects to AV company's servers. Whereas this "one-size-fits-all" solution may claim that it has showcased that it can address all issues at the same time, in practice it only brings new questions, such as: (i) is static signature matching enough for most cases or should the user turn on runtime monitoring permanently?; (ii) should runtime monitoring be enabled for all processes or only for newly-launched ones?; (iii) which fraction of suspicious objects should users outsource to the cloud inspector?

These challenges are hardly ever tackled by "one-size-fits-all" solutions, which end up transferring to users, analytics, and system administrators the responsibility to properly identifying the solution's best parameters for their use cases.

This mode of operation, where a solution attempts to accomplish many goals instead of exploring a problem in-depth is problematic because each claimed contribution is not comprehensively explored and its implications are not fully understood.

As discussed before, the feasibility of a solution for a given scenario should be backed by data from prior observational studies. For example, many userland detection solutions can potentially operate in kernel-mode. However, it is important to evaluate first to what extent the solution addresses the problem in userland before making it generic to both levels of abstraction. Similarly, whereas an analysis solution can also operate in detection mode, having it providing insights about an underexplored scenario may be more scientifically significant than operating in a "2-in-1" fashion by integrating this approach to build a detector enhanced by a marginal rate.

The scholarly work that closest investigated a side-effect of "one-size-fits-all" solutions is the Android policy evaluation by Chen et al. [41], where authors observed that access control frameworks are often ineffective because "*existing Android devices contain a wide variety of SEAndroid policies, depending on both the version of Android as well as the device manufacturer*" and even user-defined policies are not enough to prevent privilege escalation.

Pitfall 8: Developing real-time and offline solutions under the same assumptions. Engineering solutions are one of the most common types of proposed malware research. A plausible reason for such prevalence is the pressing need to protect users and corporate devices. Engineering solutions can be classified as real-time and offline, according to their analysis/detection timing approaches. In real-time solutions, the collected data (e.g., API calls) is classified or flagged as malicious as soon as it is captured (e.g., within a sliding window). Offline solutions are usually used for analysis, classify or flag an execution after **all** data (e.g., an entire trace) is captured. Each type of approach presents their own advantages and drawbacks, which in practice are often mixed, resulting in flawed designs and evaluations. Offline solutions present two major advantages over real-time ones: simplicity of implementation and whole-context view. The implementation is simpler because offline solutions: (i) do not need to concern about monitoring overhead, a constant concern for real-time so-

lutions because of performance penalties affecting users, who can affect user's experience, potentially leading users to turn off the solution; (ii) do not need to protect themselves against attacks, contrary to real-time solutions, because they operate in a protected environment; and (iii) do not need to concern about knowledge databases (e.g., signature, training model, opposite to real-time solutions which need to consider database size and updates), also because they operate in controlled environments, which no strict constraints.

In practice, despite most papers claiming the applicability of their proposed solutions in real-time, **all** of the 132 papers proposing defense solutions considered in our systematization are actually off-line detection tools because they do not present either solutions or reasoning about the aforementioned challenges, with only four papers acknowledging that. A good example of an article properly handling the differences between online and offline detection approaches is observed in the work of Khasawneh et. al [99], which not only acknowledges both operation modes (e.g., "*We also evaluate the performance of the ensemble detectors in both offline and online detection.*"), but also acknowledges their performance differences (e.g., "*the detection success offline: i.e., given the full trace of program execution*").

Challenge 5: Understanding Prototypes and Real-World Solutions. The security field is very dynamic and new solutions are often being proposed and the nature of these proposals is very diverse (see the interesting case of an academic mobile malware detector transition to the market [72]). Academic researchers tends to focus on novel proposals, whereas industry researchers usually work on developing real-world solutions. Ideally, these two types of research should be complementary, with one providing insights for the development of the other. This type of cooperation, however, requires understanding of the pros and cons of each type of proposal, which is often not clear for many researchers.

When prototyping, researchers are free to create novel concepts without the constraints of the real world and concerns about deployment. In prototype-based studies, researchers are usually concerned in presenting a new idea rather than to what extent the idea can be transitioned to practice. In general, prototyping assumption is that once the idea is validated, some third party (e.g., a system vendor) can later provide a real-world implementation for the proposed solution. As a drawback, prototype-based solutions cannot be immediately deployed for use.

Research on real-world solutions, in turn, focuses on ready deployment, thus exerting actual benefits to users, corporations, and analysts. These proposals usually rely on previous approaches and focus on practical constraints, such as storage requirements, energy efficiency, and interaction with OS and other applications. Due to these constraints, which can impose high development and maintenance costs, it is common that some aspects of the original proposal are discarded to allow for feasible implementation.

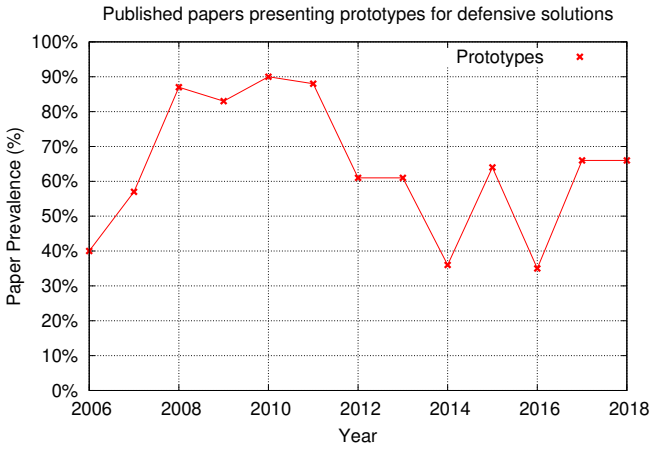

Figure 10: **Prototypes and real-world solutions.** Although most academic research focus on prototype solutions, labeling solutions as such is still not a common practice.

We propose that both types of research are important and that both should be considered equally valuable by the community. However, due to the academic community inclination of favoring highly innovative work, academic researchers tend to propose more prototypes than real-world solutions. On one hand, science and engineering can only make breakthroughs if highly innovative ideas are proposed, thus researchers should keep investing in prototypes for presenting their innovative ideas without constraints. On the other hand, actual progress and broader impacts can only be made if highly innovative ideas are transitioned to practice. Going back to the NASA analogy, what progress would have been made if no practical technologies had been developed to ***actually*** reach outer space?

Our systematization revealed that many researchers do not position their solutions as prototypes, despite none of the 132 engineering solution papers reviewed presented a solution that is mature enough to be deployed in practice. Even solutions which were further transitioned to practice (e.g., the ROP mitigation KBouncer [150] become a product [53]) could not be considered as *ready-to-market* in the time of their publication since they lack even basic functionalities such as Graphical User Interfaces (GUIs) and configuration files.

One hypothesis for such a phenomenon is the lack of formalization on the role of prototypes and real-world solutions. Figure 10 shows that from 2000 to 2005, when the field was not well established, papers were not clearly positioned as either prototype or real-world solution (even considering very lax definitions of prototypes, such as the

author's self-positioning their creations as such). As the field matured (2006-2012), most researchers positioned their contributions as prototypes. In recent years (2013-2018), the percentage of researchers positioning their solutions as prototypes has oscillated. This oscillation does not correlate to more real-world solutions being proposed, but actually to researchers not clearly labeling their proposed work as either prototype or real-world. It may indicate that researchers are facing difficulties classifying their own work or might be concerned of biases towards both types of classifications (e.g., prototypes seen as "academic exercises" and real-world solutions labeled as "incremental, engineering, or development work").

Correctly positioning their work is important for researchers because reviewer's criticisms about their work are also biased according to this positioning, as some reviewers are more prone to accept prototyping work whereas others to accept more real-world work. Positioning a solution also implies on acknowledging some development trade-offs. For instance, malware researchers are typically required to select an underlying platform to support their solutions. When supported by virtual machines (VMs), malware experiments can be easily set up and analysis procedures usually scale well, as reverting VM snapshots is enough for restoring the system to a clean state. VMs, however, can be detected by armored malware samples, thus resulting in reviewers complaining about evasion issues. Bare-metal-based solutions, in turn, are free from side-effects, avoiding sample evasion and being more suitable for the development of solutions targeting more advanced threats. As a drawback, bare-metal systems are hard to reset to a clean state, as no native snapshot support is available, limiting experiments scaling and inclusion of large datasets. Whereas analysis experiments can leverage a combination of VMs and bare-metal machines to overcome evasion, reviewers should acknowledge that researchers proposing the development of new solutions are required to opt for one of these environments under the cost of having to implement their solution multiple times only to prove their proposed concept as feasible. Researchers, in turn, must acknowledge the limitations of the selected environment and point out development gaps.

Another typical design choice that malware researchers face is the targeted OS to leverage for their solutions. An open-source OS, such as a Linux streamlines instrumentation, as its kernel can be recompiled with new structure's definitions, thus constituting a good prototyping platform for the proposal of new experiments. The Linux OS, however, presents fewer malware samples available for evaluation compared to targeting it in comparison to the Windows OS. Developing a solution for Windows, on the other hand, can be considered as aiming to provide a real-world solution, as it is the OS most targeted by attackers [7]. This OS, however, is closed source, thus not allowing kernel recompilation for structures redefinition, which limits solution scope [25]. On Windows, for instance, instead of kernel modifications, many changes must be deployed in userland, a more restrictive threat model, but compatible with a real-world solution.
If the peer-review process does not fully acknowledge this trade-off, the choice between the adoption of Linux or Windows as base for a solution development would turn into the choice about which experimental step would be considered as limited: implementation (restricted in Windows) or evaluation (restricted in Linux). Similarly, researchers working in mobile environments are required to adopt threat models that might be more or less intrusive. For example, approaches requiring jailbreaking OS native protections (e.g., Android rooting) are more comprehensive, but one may claim that their implementations are unfeasible in practice due to the vendor's security policies of not allowing device rooting. On the other hand, self-contained approaches are immediately deployable, but one might claim that these hypothetical solutions can be defeated by privileged actors (e.g., kernel rootkits).
Further, we identified a possible conflict between prototypes and real-world solutions in the emerging field of hardware-assisted security, which encompasses both malicious codes exploiting hardware vulnerabilities [191, 67] as well as the development of hardware support for software protection [24]. Whereas hardware is often designed using simulators (e.g., Intel PIN [128]), security evaluations are usually expected to be performed in real systems (e.g., exploiting a real vulnerability). In addition, as malware research is multi-disciplinary, reviewers from distinct fields (system security vs. computer architecture) might naturally exhibit different biases and preferences according to their working fields standards (see biases in computer architecture research [109]). Therefore, researchers in the field might expect some reviewer's feedback sometimes complaining more about the feasibility of the prototype whereas others will complain more about the security evaluation. For instance, computer architecture experts tend to be more prone to accept prototyping, as this community is more used to the challenges for modifying actual processors and often assume that vendors can better transition solutions to practice [73]. Security experts, in turn, tend to be more prone to question the viability of vendors adopting the proposed solutions due to the practical nature of most security research work.

4.4 Experiment Design

As for the design of solutions, pitfalls originated from unrealistic scenarios also appear in experiment design, as following discussed in this section:

Pitfall 9: Introducing biases in definition and evaluation of datasets. To perform experiments in

a significant scenario, researchers should balance their datasets to avoid biases, i.e. experimental results being dominated by a subset of all possibilities. Researchers conducting experiments involving machine learning classification are particularly concerned with dataset biases. For example, they do not want a single malware family (e.g., Trojans) to dominate other malware families (e.g., worms, downloaders, bankers, etc). To avoid family representation biases in malware experiments, researchers strive to define datasets with equally represented malware samples counterbalanced by family type. Whereas such choice seems reasonable, it also introduces biases because equal representation of samples implies that all scenarios (end-users, corporation sectors, countries) are equally targeted by a balanced set of malware families, in an "one-size-fits-all" fashion (see Section 4.3 for another example where the "one-size-fits-all" pitfall applies). In practice, no environment is known to be equally targeted by all malware families. On the contrary, some environments may present unbalanced family prevalence, such as the Brazilian scenario, which is dominated by banking malware [35]. Therefore, targeted classifiers can potentially outperform their "one-size-fits-all" counterparts for a particular scenario. Unfortunately, most malware research does not discuss this assumption and also does not compare classifier results considering multiple datasets having distinct family distribution.

Also, users are more prone to be targeted (and infected) by malware distributed via phishing messages than by automated worms [63], thus showing that purely technical aspects (e.g. dataset with families equally represented) has been trumping key cultural and environmental aspects pertaining to the audience of the solution. Ideally, a solution targeting a given scenario should be evaluated with a dataset reflecting the characteristic of that scenario. Unfortunately, there is a scarcity of studies covering particular scenarios (see Section 4.1, such as specific countries, industry sectors, which makes the development of targeted solutions harder. Among all defensive papers, only seven discussed dataset distribution and its relevance to the scenario where the solution should operate. Stringhini et al. [183], for instance, proposes a system "*able to detect malicious web pages by looking at their redirection graphs*" and explain that their evaluation matches real-world requirements because their evaluation dataset was "*collected from the users of a popular anti-virus tool*".

Pitfall 10: Falling for the Anchor bias when defining datasets. Defining an appropriate sample dataset (malware and goodware) is key to most[3] malware research. A dataset size too small might not be representative of a real scenario, thus characterizing results as anecdotal evidence. Extremely large datasets, in turn, might end up proving almost anything, given that even statistically-rare constructions appear in the long tail, but these might not be prevalent in any actual scenario, thus limiting the application of researcher's discoveries as the expected conditions would be never met.

Ideally, to define a good dataset, researchers should first identify the characteristics of the environments in which their solutions are designed to operate, by leveraging samples targeting such environment to avoid introducing biases (see Section 4.4 for a more comprehensive discussion). This two-phase requirement highlights the differences between observational studies and engineering solutions(see Section 4.1). Whereas the first type usually requires a large number of samples in the evaluation process for appropriate environment characterization, the second type can potentially leverage fewer samples once previous studies have shown that the considered samples are appropriate for the environmental characteristics and present a significant threat model (see Section 4.2). As pointed by Szurdi et al. []: "*Investigating... behavior longitudinally can give us insights which might generalize to traditional cybercrime and cybercriminals*". Another important factor in the dataset definition is the type of research being conducted regarding targeted OS (Windows, Linux, Android) and approach (dynamic vs. static). Windows and Android environments provide researchers with many more samples than Linux. Further, static approaches can process a substantially larger number of samples per unit of time than dynamic approaches.

Because of the paucity in observational studies and lack of dataset definition guidelines, researchers end up establishing datasets in an ad-hoc manner, adding challenges to the peer-review process. More specifically both researchers and reviewers end up falling for the Anchor bias [64], a cognitive bias where an individual relies too heavily on an initial piece of information offered (the "anchor") during decision-making to make subsequent judgments. Once the value of this anchor is set, all future decision-making is performed in relation to the anchor. Whereas this effect is present in many research areas (e.g., forensics [184]), its impact on malware research is particularly noticeable. For example, consider a paper proposing a new method to classify malware for Windows using static methods and adopting a dataset with one million samples. After publication, one million samples implicitly become an anchor. Then, consider a researcher proposing a novel real-time (dynamic) Linux framework to detect malware via machine learning. Because the approach leveraged Linux (fewer samples available) and is dynamic (i.e., requiring more time to run samples, prepare the environment for samples, etc.), it will be nearly impossible for this proposal to meet the "anchor requirements": a dataset with one million or even hundreds of thousands of samples. Next, after peer-review, it is plausible to hypothesize that the proposal might receive feedback pointing out the use of a "small" dataset.

[3]Offensive papers will present distinct requirements

Figure 11 shows dataset size distribution for all defensive and observational malware research papers published since 2000. As hypothesized, no pattern can be identified in such distribution, with published papers presenting both very small and very large dataset sizes in all years. As a result, the malware research fields tends to become completely fragmented, which implies difficulties to develop the field in a scientific way as no standard practice is established.

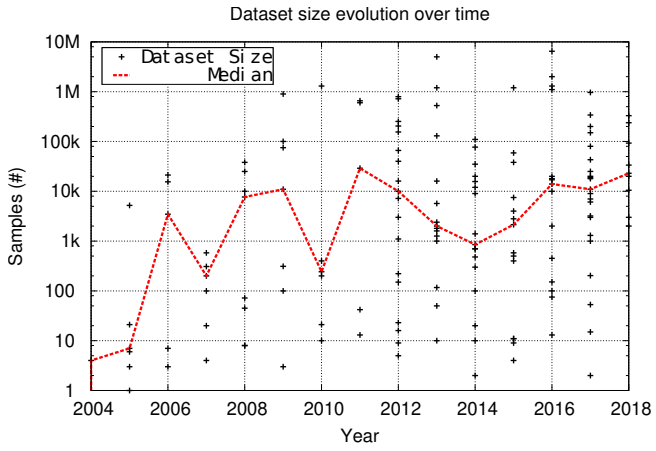

Figure 11: **Dataset size over time.** Whereas the median number of considered samples has been continuously growing, the dataset size definition is still an ad-hoc decision, thus resulting in non-uniform evaluations.

Figure 11 also shows a growth both in the frequency of papers evaluating very large datasets (million samples) and in the median dataset size over time, indicating the occurrence of the Anchor effect. This bias should be avoided by the community when aiming to develop malware research in a stronger scientific field under the risk of presenting contradictory results, such as a paper claiming that 900K samples are enough to present a **landscape of all** malicious behaviors [13] and another one claiming that more than 1M samples are required **only to train** its detection procedure [86].

Table 4: **Dataset size by platform.** Some platforms have more samples available than others, thus affecting dataset establishment.

Platform	Minimum	Median	Maximum
Windows	1	2.1K	6.5M
Android	2	10K	2M
Linux	3	72	10.5K

Relevant to this discussion, Herley and van Oorschot [82] suggested that the security community "*stop insisting that quantitative is better than qualitative; both types of measurement are useful*". We propose that dataset definition decisions consider environmental and context aspects in addition to other currently used criteria (e.g., considering only the number of samples). The importance of context for dataset size evaluation is illustrated in Table 4.4, which shows the clear difference between the minimum, median and average dataset sizes considered in papers targeting distinct platforms. Studies targeting Android present a dataset size median (10K) greater than studies targeting Windows (2.1K), despite Android being a relatively newer platform compared to Windows. This can be explained by the higher availability of apps for Android (malicious and benign), including both legitimate software present in the apps stores, as well as malware samples targeting mobile device users. The consolidated Windows research reflects in its largest research dataset (6.5M) face to the largest Android one (2M). When considering network traffic studies, the number of evaluated malware samples grows up to ≈27M [120].

Another observation is that malware studies targeting Linux present, as expected, both the lowest median (72) and lowest maximum dataset size (10.5K) values. The natural reason is Linux platform being less popular than Android and Windows, thus, being less targeted by malware writers. Therefore, it should not be reasonable to expect a Linux proposal to use sample sizes comparable to a Windows or Android solution, thus reinforcing our claim for considering contextual aspects in dataset size definition procedures.

An example of a representative dataset despite its size is the one presented in the Control-Flow Jujutsu attack [31] (offensive research), which is exemplified with a single exploit and demonstrated its impact to the whole class of Control Flow Integrity (CFI) solutions based in the `CALL-RET` policy.

Pitfall 11: Failing to develop uniform and standardized malware and goodware repositories. A significant challenge for defining a dataset for malware research is the sample collection procedure, mainly due to the lack of uniform or standardized repositories, which often results in introduced biases and limited evaluations.

Figure 12 shows that, for most current malware research, malware samples have been retrieved mainly using one of the three following methods: honeypots, blacklist crawling, or private partner data sharing. These sources present distinct drawbacks for malware experiments. For example, honeypots cover a limited attack surface, only capturing samples targeting the same platform as the honeypot platform and in the same network, thus often yielding the lowest collection rate among all the three methods. Blacklists usually yield a good number of samples, but sample quality (e.g., diversity, age) is dependent on a particular user's contributions for updating the list and/or repository, thus offering the analyst no control over the available samples. Private partners' repositories usually present a relevant and comprehensive amount of information about the collected samples, which typically cover a real scenario, thus explaining their prevalence in

network traffic research (Figure 13). However, due to their private nature, (e.g., information shared by ISPs), research-based on such repositories are often hard to reproduce, as malware and traffic samples are almost never shared with the community and their descriptions are usually limited so as not to disclose much partner information, sometimes even omitting the partner name itself. We were able to find multiple occurrences of this practice in the academic literature: (i) "*was installed at our partner ISP*" [38]; (ii) "*botnet C&C servers provided by industry partners*" [201]; and (iii) "*From a partner we acquired a list of nodes*" [152].

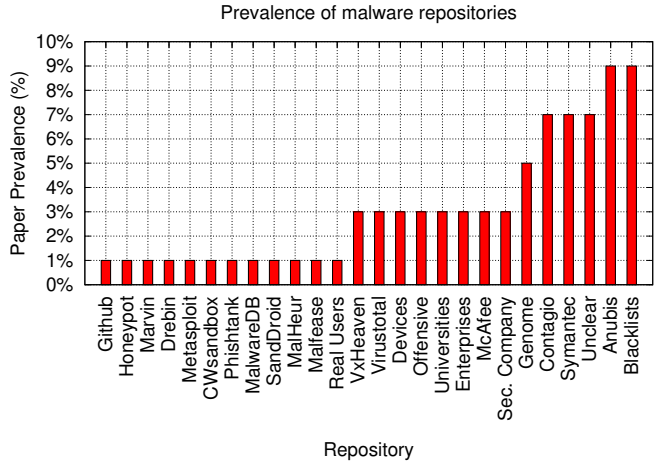

Figure 12: **Considered Malware repositories in the entire period.** Most research rely on blacklists, private or custom repositories.

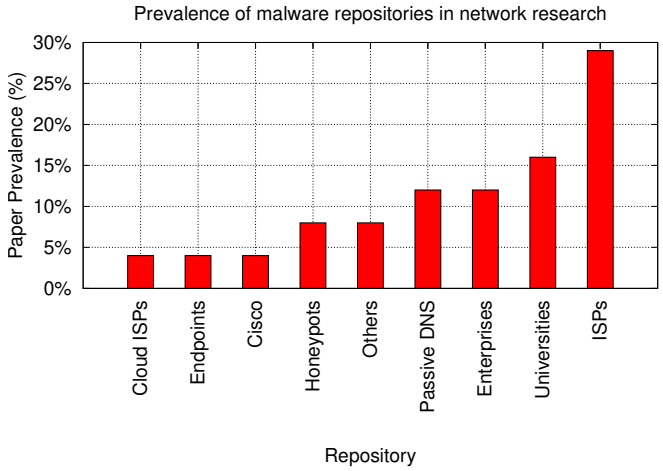

Figure 13: **Network repositories.** Most research rely on data shared by private partners.

A common drawback of most (if not all) malware repositories (noticeably blacklists and public repositories) is that they are polluted, i.e., present other data in addition to the malicious samples, such as misdetected legitimate software and corrupted binaries (e.g, binary objects derived from failed attempts to dump samples from infected systems). For instance, by searching Virustotal's database, one can easily find artifacts triggering wrong detection results [196], such as innocuous code excerpts or binary blobs submitted by researchers (and even attackers) as part of their detection evaluations. As these object's codes are unreachable or present incorrect endianness, they are not useful for analysis purposes. To create an appropriate dataset, such objects must be discarded to avoid introducing bias in analysis procedures. Unfortunately, none of the reviewed papers described this step, which prevented our present analysis to identify whether this step is implicitly assumed or simply disregarded by the researchers.

Another drawback of many malware repositories is that they also store unclassified samples, which makes it hard for researchers to identify families or the platform the sample is targeting. Further, many repositories and blacklists are unreliable (e.g., become offline after some time), which adds obstacles to research reproducibility.

Even when researchers can classify the malware samples from a given repository using some other method, they quickly realize that most malware repositories are unbalanced, i.e., some malware families are more prevalent than others. More specifically, user-dependent repositories will be biased by the malware samples targeting the top contributing users. Similarly, sample platforms are biased by their market share (e.g., Windows vs Linux, desktop vs mobile), which makes it harder, for instance, to collect a significant amount of Linux malware than Windows samples. Ideally, research work should acknowledge these biases, as done by Le Blond et al. [20]: "*As a result of these biases, our VirusTotal dataset offers a partial coverage of attacks where individuals and NGOs are likely over-represented.*"

Another challenge is the lack of sample identification date, which places obstacles to the conduction of longitudinal studies. Some proposals try to overcome this challenge by making assumptions, such as considering VirusTotal submission dates as sample creation date [85, 101]. This assumption can be misleading, as it implicitly depends on AVs capacity of detecting the samples. Therefore, when considering sample's creation date as the same as sample submission date, researchers might not be evaluating when samples were created, but actually when AVs detected or were notified about them. This lack of proper temporal labelling affects research focusing on sample evolution issues, such as when machine learning-based malware classifiers start experiencing concept-drift [95, 35].

Pitfall 12: Assuming crawled applications as benign without validation. The collection of benign software (goodware) to be used as ground truth for security evaluations is also challenging. and the selected samples directly affect evaluation results. Fig-

ure 14 shows that most papers proposing defense solutions rely on crawling popular application repositories (e.g., `download.com` for desktop and `Google Play` for Android). After raw collection, researchers need to ensure that the applications are indeed benign and representative of a given scenario.

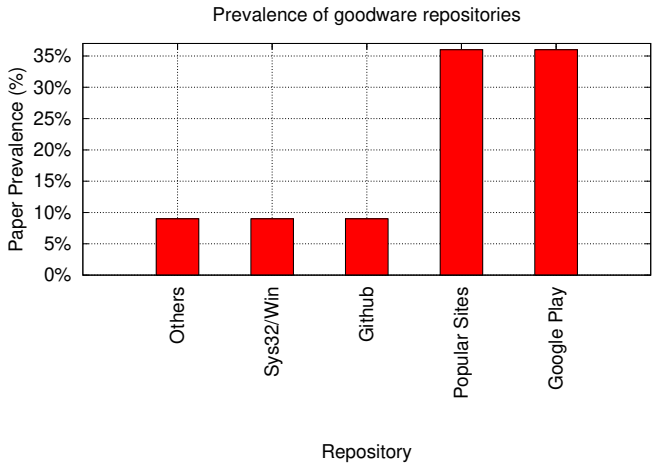

Figure 14: **Considered goodware repositories in the entire period.** Most research rely on crawling popular application repositories. Downloaded applications are not guaranteed to be benign.

One important consideration when defining goodware datasets to assure the effectiveness of a proposed security solution is including common applications, i.e., those that are *actually* installed and used by users belonging to the environment the solution is supposed to operate (see Section 4.4). However, the current trend of leveraging larger malware datasets creates, via Anchor bias (see Section 4.4), expectations of comparable large and counterbalanced goodware datasets. However, when leveraging a large number of applications from software repositories, researchers risk considering as ground truth applications that are not common (e.g., listed in the 100^{th} top downloaded apps page).

In addition to representativeness issues, considering applications crawled from popular repositories as a ground truth for goodware might also cause a solution to include considering malicious entities as legitimate, as these may be embedded in Trojanized applications (e.g., an installer embedding an adware [22]) in the dataset. Whereas many reputable websites and app stores scrutinize their distributed payloads, the benign characteristics of a dataset cannot be ensured without proper inspection. In all defensive solution proposals considered in this work, only 15 ($\approx$ 10%) explained whether and how they filtered goodware samples (e.g., Willems et al. [202] stating that "*in 3 cases one or more supported scanners returned a positive result. We checked those samples by hand and did not find any malicious content within them.*").

Pitfall 13: Evaluating the application's installers rather than the actual application's components. Another problem of considering applications crawled from popular software repositories in research works is that applications are usually distributed in the form of installers, and not as standalone applications. Installers usually compress all application components (e.g, binaries, libraries, drivers, so on) into a single component that does not present the same characteristics of any of the embedded components individually, which might result in research pitfalls if not methodologically handed.

For static procedures, the problem of considering application installers is that the compression of files in a single component might hide strings, imports, and resources present in the embedded components from the inspection procedures. This might result, for instance, in "trojanized" components of an originally benign software being hidden from the evaluated security component, thus influence the evaluation results.

For dynamic procedures, the challenge is to trigger the target application behavior, since the application must be first installed. The installation procedure must be automated, which is usually done via clickers [22]. If this task is not considered, the proposed security solution will be evaluating the installer application rather than the target one. This might significantly bias evaluations since the characteristics of an installer execution are completely distinct from the characteristics of the installed applications. For instance, installers will likely interact more with the filesystem (to drop the files to the correct folders) than to interact with other system processes, as done by malware samples.

None of the papers we investigated that considered applications crawled from software repositories specified whether they considered the installed applications or the installers, and how the installation procedures were automated. Whereas this does not imply that their experimental strategy is wrong, we consider that reporting these factors is a good practice for the evaluation and reproducibility of malware research work.

Pitfall 14: Evaluating real-time and offline solutions under the same criteria. Offline solutions present a significant advantage over their real-time counterparts: they have the entire trace available to make decisions, which naturally increases their detection rates. Real-time solutions, in turn, usually have to make decisions on-the-fly with less data (e.g. sliding windows of data), a partial view from the system or, API calls collected until the inspection moment, which naturally limits their decision capabilities and increases their FP rates. As a drawback, due to their nature of considering the whole-data view, offline solutions applied to a real-time scenario would only be able to detect a malicious sample after the sample performed all its malicious ac-

tions (e.g., data exfiltration), and thus compromise the system.

Thus, whereas offline solutions are evaluated almost exclusively by their detection rates, the evaluation of real-time solutions should also include additional criteria. An important metric for a real-time solution is the detection delay, i.e. how long it takes to flag a sample as malicious: the earlier a threat is detected, the better, as the threat is blocked before causing more harm to the system.

Despite detection delay is an important metric, a real-time solution cannot adopt an aggressive policy and start detecting any sample as malicious whenever it does not have enough data to decide, as the false positives (FP) rate is also an important metric. Whereas in offline solutions a FP does not affect user experience, in real-solutions, it might prevent a user to interact with a legitimate application. In addition, the differences in the amount of available data at decision time also affect the classifier's training strategy. Whereas classifiers for offline approaches can be trained with entire traces, classifiers for online solutions should be trained with multiple sliding windows and strides derived from entire traces to reflect their partial view. An online solution trained using complete traces would have a very low matching score during runtime operation due to the lack of enough data to be compared.

Unfortunately, detection solutions still do not clearly distinguish these real-time and offline differences. None of the evaluated work clearly pointed out the online monitoring challenges that they proposed to tackle nor evaluated the detection delay. Therefore, we can still observe unreasonable claims, such as real-time classifiers based on complete malware traces. For instance, Kwon et al. [111] proposes an online detection method ("*we perform an experiment where we simulate the way our classifier would be employed operationally, for detecting malware in an online manner.*") and states that "*We train the RFC on the training set, with the same parameters used on section 5.4.*" (when all samples were considered in an offline manner) and "*Then we apply the trained classifier on the testing set. We obtain 99.8% TP rate*". Despite achieving a high accuracy result for the presented experiment, it is unlikely that the same parameters for offline and online detectors would lead to the best accuracy rates when operating in an actual scenario.

Pitfall 15: Evaluating analysis and detection approaches using the same criteria. More than different goals, as presented in Section 4.1, analysis and detection work also require distinct approaches for their evaluations, and understanding these requirements is important to properly develop and review these experiments. Unfortunately, we can still identify research work mixing analysis and detection concepts.

Analysis procedures aim to collect data for further research steps (e.g., drawing a landscape, training a classifier, so on). Despite all challenges for tracing in-the-wild collected malware (see Section 4.5), analysis studies usually do not suffer too much with these issues because they usually require only recording API calls attempts for characterizing a malicious behavior, although the malware couldn't successfully complete the requested action.

Detection approaches, in turn, involve actually observing a successful sample run. evaluate detection techniques and unlike analysis studies, just monitoring API calls despite their result is not enough for evaluating the proposed solutions. Ideally, they must ensure that the samples effectively run and trigger their detection invariants, which can be challenging. Given the challenges for reproducing malware experiments (see Section 4.5), such as samples requiring legacy libraries to run, detection experiments should require analysts to first reverse engineer the collected malware samples to identify their execution requirements and later select the ones which successfully executed, which naturally limits experiment scale, as it implies on a limited dataset size due to the required manual interaction. To overcome the dataset size challenge, detection studies could leverage fewer samples than analysis ones once an analyst could prove that the considered dataset is representative of a real scenario, which can be done, for instance, by referring to previous studies, thus our claim of the importance of these observational studies.

Unfortunately, most researchers do not describe whether/how they reversed samples before testing their solutions (among all considered detection papers, only 13 (≈8%) acknowledged reversing samples before testing), which did not allow us to identify whether malware execution failures were due to the claimed solution detection effectiveness or to missing components required for execution.

Pitfall 16: Using non-reproducible methodology for defining dataset and designing experiments. Reproducibility is a key aspect to define some investigation as scientific, since it allows other researchers to disprove or confirm previous results, thus making predictions and advancing the field. However, reproducibility is hard to achieve due to both practical and theoretical issues, and acknowledging reproducibility limitations helps (i) preventing other groups from spending time attempting to reproduce limited experiments steps; (ii) shedding light on the shortcomings of reproducibility of certain types of research; and (iii) motivating researchers to advance reproducibility in further research work.

In practice, many malware research derives from data shared by private partners, leverage proprietary solutions, or depend on non-disclosure-agreements, which prevents researchers from releasing their datasets. Among all considered papers using datasets, only 33 (≈ 7%) released their datasets (e.g., malware binaries or net-

work traffics), showing that only a small portion of them is reproducible.

More than having access to a dataset, reproducing malware research is also challenging due to theoretical limitations. For example, non-determinism at OS and environment levels can yield different execution outcomes for a given malware sample. This phenomenon is more frequently observed in network-based malware experiments. For instance, modern botnets often contact domains at runtime using a Domain Generation Algorithm (DGA), which results in distinct DNS queries and contacted IP addresses along with their multiple runs. There is no guarantee that the execution outcome of a given sample will be the same when run by distinct research teams on different occasions. Further, even for researchers that can track network communications, there is no guarantee that malware's C&C servers will be always available (e.g., became sinkholed). Acknowledging these issues is particularly relevant for researchers trying to reproduce experiments with samples from previous studies, because their C&C may have been sinkholed. In this case, samples would fail to retrieve their malicious payload and prematurely abort their executions, thus presenting smaller execution traces in comparison to their original reports.

Pitfall 17: Comparing Apples to Oranges. It is not unusual for proposals to compare their evaluation results with those from prior literature tackling the same problem, for instance, comparing the accuracy of two detection approaches involving machine learning. Such comparison, however, should be carefully performed to avoiding misleading assertions.

As a consequence of the lack of standard repositories, many works end up comparing their evaluation results (e.g., classifiers accuracy) with other values reported in the literature. Whereas comparing work seems to be straightforward, authors should care to perform fair evaluations, such as comparing studies leveraging the same datasets, thus avoiding presenting results deemed to outperform literature results but which do not achieve such performance in actual scenarios.

As a didactic analogy, consider image classification challenges, whose objective is to identify objects represented in images (e.g., buildings, animals, locations, so on). The challenges often provide multiple datasets. For instance, the CIFAR challenge [110] is composed of two datasets: CIFAR-100, which has one hundred classes of images, and CIFAR-10, which is a filtered version of CIFAR-100, containing just ten classes. Imagine two research work proposing distinct engineering solutions for image classification, one of them leveraging CIFAR-10 and the other leveraging CIFAR-100. Although one of the approaches present a higher accuracy than the other, is it fair to say that this one is better than the other? Clearly not, because the task involved in classifying distinct classes is also distinct. The same reasoning is valid for malware research, especially those involving machine learning. Therefore, authors should care to not perform comparisons involving distinct classes of applications, such as comparing, for instance, approaches involving Dynamic System Call Dependency Graphs, a computationally costly approach, with static feature extraction approaches is misleading because each type of work presents different nature and challenges.

4.5 Test of Hypothesis/Evaluation of Solutions

In addition to theoretical issues regarding the research and solution design steps, research pitfalls may also originate from practical aspects, even when experimental procedures are properly defined, for example when leveraged tools for data collection and analysis present (inherent or technological) limitations, which are often not well understood and acknowledged.

Pitfall 18: Using broken, raw samples. Many research work in computer science and engineering leverage some type of dynamic analysis technique to inspect and characterize applications (e.g., computer architecture papers profiling branch prediction rates). In common, all these research work present an implicit assumption that all samples are well-behaved and self-contained, thus running without problems in a sandbox solution.

Many malware research work uses dynamic analysis techniques to study samples and/or evaluate the effectiveness of defensive solutions. Unlike the computer architecture example on profiling branch prediction rates, the sandbox execution feasibility assumption is sometimes flawed, given peculiarities of malware samples when compared to standard applications. For example, while common applications (e.g., browsers, text-editors) are self-contained, modern malware (noticeably downloaders and droppers) are composed of multiple sub-modules, which makes their analysis challenging, as such modules can not always be captured, allowing a holistic analysis of the sample. Moreover, these sub-modules often present inter-dependencies, which requires analysts to guess the correct execution order of samples (e.g., loader, downloader, and persistence modules). Another challenge is getting access to loaders (for infection launch) and libraries (including proper version), required for sample successful execution. This also makes infection launch harder because most of times analysts do not have malware loaders, which are required for injection of malicious payloads in their target processes, and also for launching malware samples with proper command line arguments. Modular malware execution is also challenging because the libraries that they require to run may become outdated in current systems, thus requiring analysts to install a previous library version in current systems under the risk of prematurely aborting sample's execution due

to version incompatibility. Further, shared libraries (e.g. Windows DLLs) may fail to execute in automated analysis systems because of the need to manually identify function entry points.

Unfortunately, none of the papers leveraging dynamic analysis in our systematization described how they handled such cases, which prevented us from discovering why these cases are not being reported, either because they were explicitly disconsidered from the evaluation procedures or whether these aspects are being overlooked by the community.

Pitfall 19: Failing to establish criteria for assessing sample execution in a sandbox. Execution of malware in sandboxed environments brings many challenges. When a sample runs in sandbox, even having a standard entry point, there are no guarantees that execution was actually successful because the sample could have applied anti-analysis techniques [194], or failed due to multiple reasons, such as corrupted samples or OS' incompatibilities.

Consider, for instance, an execution trace generating only a few API calls. After sample execution the following questions arise: (i) are these APIs calls the only ones supposed to invoked by the sample?; (ii) was the execution aborted prematurely and the observed APIs calls were just system cleanup routines?; or (iii) did the sample evade analysis?

Therefore, establishing criteria for sample successful execution in sandbox (e.g., minimum number of invoked API calls or exhibited behaviors) is crucial. Unfortunately, none of the engineering, defensive papers considered in this study that leveraged sandboxes presented either criteria for a successful execution of samples in sandboxed environments or percentage of samples that effectively executed. We identified an example of a clear sandbox criteria in the network study of Lever et al. [120], which explicit that their study "*excludes samples without any valid or successful DNS resolutions.*"

Only recently researchers started to systematically investigate how much the distinct sandbox execution timeouts affect malware analysis results [112]. We expect this type of analysis to be considered in future malware research work to better support experiment design decisions.

Pitfall 20: Blindly relying on AV results as ground-truth. When one thinks of malware detection, Anti-Viruses (AVs) immediately comes up in most people's minds, as AVs are still the main defense line against malware in all types of environments and given such importance, AV research brings together academia and corporate researchers, mixing prototyping and real-world solutions (see Section 4.3), resurfacing the issues related to misunderstandings of the challenges and limitations of each type of work.

Many proposals rely on AV results as ground-truth for their experiments ($\approx$ 23% of all papers considered), either for identification of sample families or for comparison of detection rates. Consequently, understanding the implications of using AVs as ground-truth is essential to understand research results.

The first challenge researchers face when relying on AVs is that nobody really knows how commercial AVs work. Whereas detection procedures such as signature matching and heuristics are described in the literature, nobody is able to identify which of these methods was applied (and succeeded). When an AV solution reports a file as malicious, a researcher is not informed about what specific methods contributed to this diagnosis (e.g., signature matching, heuristics, or a combination of methods), which makes experiments considering AVs as ground-truth challenging. Consider a new pattern matching mechanism proposal, which reportedly performs 10% worse than a given AV. Most would consider the impact of this solution a small impact, thus not advancing the state-of-the-art. However, the AV results might be based on the use of a combination of detection approaches, such as multiple heuristics and signatures, which makes the comparison unfair. If the pattern matching engine of AV could be isolated, researchers could discover, for instance, that the new solution outperformed the commercial AV static detection in 100%. As an example of this scenario, consider the evaluation of the new signature schema proposed by Feng et al. [66]. Their evaluation states that "*VirusTotal agreed with ASTROID on each of these 8 apps*", achieving the **same** results as commercial AVs. However, since we have no guarantees that Virustotal's AVs leverage only signatures, the real conclusion might be that this approach **outperformed** commercial AV results.

Therefore, we advocate for the development and use of more configurable AV solutions for the development of more scientifically rigorous studies. While requiring commercial AVs to adopt additional configuration schemes is unrealistic, the community could set expectations for AV companies practices such as providing detection results metadata, so that researchers can cluster the samples detected using the same technique. We acknowledge that many AV companies would not be inclined to adopt such proposal because of intellectual property issues. Alternatively, an interesting future work for the field is the development of standardized AV evaluation platforms, such as an academic, open-source AV which could be easily instrumented for performing malware detection experiments.

We highlight that while there are open source AV initiatives (e.g., ClamAV[49], they do not resemble a fully-commercial AV, thus not being suitable as ground-truth for malware detection experiments.

The impacts of the lack of understanding about AV's inner working are even more noticeable when one considers

that commercial AVs do not follow a standard operation model. Therefore, distinct AVs may produce different results, even when evaluated with the same dataset. A noticeable example of such non-uniformity is samples labeling, where each AV solution follows their own rules and adds internally created extensions to sample's labels. This non-uniformity makes research reproduction hard, as a dataset labeled by one AV (e.g., all samples are Trojans) cannot be compared to another dataset having the same labels but attributed to another AV, as nobody knows how the first AV would label these samples. In practice, the literature has already demonstrated that considering AV labels for sample classification may even decrease classifier's accuracy [30]. To overcome this challenge, recent research has proposed AVClass, to normalize AV labels [171]. Whereas this proposal addresses the non-uniformity issue, only $\approx 33\%$ of papers using AVs as ground-truth published after AVClass release adopted such normalization procedure.
Finally, due to the lack of understanding about AV's internals, AV feedback, in general, is limited. Although AV companies periodically release reports, these publications cannot be interpreted as technically sound to drive research. Academic studies have already shown that, in practice, AV reports do not expedite malware infections clean up [192].

4.6 Summary

Once we have discussed all challenges and pitfalls in details, we now recap the the most important findings of our literature review and analysis (in no specific importance order).

1. **Inbalance in research work types**, with more engineering solutions being proposed than any other of kind of study.
2. **Solutions developed not informed by previous study's data**, which derived from the lack of observation studies and make solutions application to real scenarios harder.
3. **Most work still don't clearly state threat models**, which limits their positioning among related work and complicates the evaluation whether they achieved their goals or not.
4. **Failure in positioning work as prototypes or real-world solutions**, which complicates evaluation and future developments attempts.
5. **Offline and online solutions developed and evaluated using the same criteria**, which leads to impractical solutions and unfair comparisons.
6. **No dataset definition criteria**, with authors and reviewers defining suitability on an ad-hoc manner, which tends to lead to an **anchor bias** towards previously published work.
7. **Few attention to dataset representativity**, with few work discussing the population targeted by the considered malware samples.
8. **Most studies are not reproducible**, either due to the use of private datasets or the absence of a list of considered malware sample's hashes.
9. **Sandbox execution criteria are not explained**, which makes hard to understand if the samples really executed or evaded analysis.
10. **Non-homogeneous AV labels are still a problem**, with distinct AVs labeling samples distinctly (in a non-comparable manner) and with researchers not performing homogenization procedures.

5 Moving Forward

In this section, we propose guidelines based on the discussed challenges and pitfalls for multiple stakeholders to advance the state-of-the-art of the malware research field.

5.1 The Field

- Increase discussions about experimentation practices on the malware field to enhance research outcomes quality. Existing venues such as USENIX CSET [189] and NDSS LASER [145] might work as a forum for discussing dataset creation and experiment designing guidelines.
- Create incentives for the development of more observational studies and offensive research to provide foundations for sound anti-malware solutions. Despite the currently non-ideal prevalence of engineering solutions, the community has already stepped in to address this drawback via targeted venues which acknowledge the importance of this type of research for cyber security, such as the `USENIX Workshop On Offensive Technologies` (WOOT), supporting offensive research. In fact, most of offensive research considered in this paper was published since 2008 in such venue, thus highlighting its positive impact on the field. Similarly, support of future workshops on observational landscapes studies is warranted to help help addressing this challenge.
- Consider academical and real-world expectations when evaluating engineering solutions, thus allowing academia to provide more efficient approaches to practical solutions adopted by the industry, such as proposing new, more efficient signature-based approaches that are still leveraged by AV solutions despite academic advances towards behavior-based detection.

- Develop classifiers for imbalanced datasets is essential to allow development of security solutions addressing actual scenarios, where equal distribution of malware families is nonexistent.
- Understand the impact of social and cultural aspects when developing anti-malware solutions for users protection. In this sense, we consider that the recent growth of the usable security field as a promising way to bridge this gap.
- Create standardized repositories and establish guidelines for dataset definitions is essential to move the community towards a more methodologically strong discipline. Notice that we are not claiming for the development of a static collection of samples, but to the development of an structured manner to handle dynamic collections of malware samples. In this sense, we currently envision attempts towards this direction in the IoT scenario [98]. Whereas this initiative does not solve current issues of existing repositories, it is an important initiative to not repeat errors from the past in the development of new technologies.

5.2 Researchers

- Clearly define the Research Objective according to one of the types of malware research (e.g., Engineering Solution, Observational/Analysis/Landscape Study, Offensive Research, Network Traffic) to streamline execution of the Malware Research method, specially regarding to proper evaluation.
- Define threat models based on real-world needs to increase research applicability and impact.
- Clearly state engineering solution's requirements to allow for adoption of proper metrics in evaluation.
- Position your solution as on-line or offline, thus easing solutions evaluation and comparison.
- Position your solution a proof-of-concept prototype or ready-for-deployment to incentive other researchers to contribute to its advancement and enhancement. The application of software maturity level assessment procedures [60], as leveraged by software engineering research, might provide criteria for researchers better positioning their solutions.
- Define datasets representative of the environment the real scenarios in which the solution is intended to operate.
- Rely on previous landscape studies insights to develop solutions and define datasets.
- State assumptions about malware and goodware samples repositories to allow biases identification and research reproducibility.
- If making comparisons to prior work, avoid simply referring to their reported results, but rather reproducing their experiments using the same dataset and methodology.
- Scan all samples, even those labeled as benign, to avoid introducing errors in ground-truth definitions due to, for instance, trojanized applications.
- Report AV detection results (e.g., sample labels) in a uniform fashion to make studies comparable (e.g., using AVClass).
- Make your datasets (binary files, hashes, execution objects) publicly available to facilitate research reproducibility.
- Make sample's execution traces publicly available to allow research reproducibility even when C&C's are sinkholed.
- When characterizing datasets, report number of samples effectively analyzed and which criteria were considered for detecting/classification of succesful execution.
- Avoid using generic AV detection results as ground-truth whenever possible to allow fair detection solutions comparisons, thus opting for more qualified detection information labels.

5.3 Reviewers

- Evaluate each work according to their stated goals: prototype vs. readily deployable solution, static vs dynamic analysis, offline vs real-world, thus acknowledging the importance of observational/landscapes studies and offensive security as basis for the developments of sound anti-malware engineering solutions.
- Support observational/landscapes studies and offensive security as basis for the developments of sound anti-malware solutions.
- Evaluate threat model fitness to real-world needs in addition to hypothesized threats described in the literature.
- Be mindful of Anchor bias when evaluating dataset size, prioritizing how the researcher defined and evaluated the representatives of the dataset for the context proposed solution is supposed to operate (corporate environment, home, lab, etc.).
- Engage in the exploratory feeling is essential to overcome the bias of claiming for more contributions at the charge of in-depth investigations, thus avoiding the risk of claiming that a solution is limited when it really solves part of a major problem.

- Understand proposals as prototypes and not as end-users solutions is essential to stimulate researchers to propose their ideas in a free way.

5.4 Conferences, Journals, and Workshops

- Support more observational/landscape and offensive security work via creation of special tracks, new workshops, and explicitly inviting in call-for-papers such line of research, as already done for Systematization of Knowledge (SoK) papers in some venues [179, 188]. and offensive security work as strong contributions in conference/journal/workshop evaluation procedures, having specially designed criteria for the evaluation of this type of work, as already done for Systematization of Knowledge (SoK) papers in some venues [179, 188].
- Adopt special landscape study sessions as part of conferences Call For Papers (CFPs), to motivate the development of this line of research, as some venues have already done regarding SoK and Surveys [179, 188, 1].
- Support more practical aspects of malware research, especially broader impacts in user and society, is essential to integrate the academical knowledge to real user's needs. In this sense, we consider that the malware scenario may learn from experiences from related security fields, such as the `Real World Crypto` conference [87], which focuses on practical aspects of cryptography. an academic conference focused in practical aspects, thus streamlining the science of implementing real-world security solutions.
- Create incentives for dataset release for paper publication, for instance, including it as one of the criterion during peer-review is a requirement that conferences and journals could adopt to push authors towards developing more reproducible scientific work. In this sense, we consider as positive initiatives such as the `NDSS Binary Analysis Research` (BAR) workshop [199], which released all datasets described in the published papers.

5.5 Industry

- Security companies: include Indicators Of Compromise (IOCs) in all threat reports to detection statistics to provide the malware research and development community with better technical information about the identified threats.
- AV companies: add detection engines and methods as part of AV labels to allow researchers to better identify how threats were detected and better evaluate their solutions. We consider that displaying the AV detection subsystems in OS logs, as performed by the Windows Defender logs [135], is a good first step towards a long journey.

6 Related Work

This paper intersects literature on improving research practices and theoretical and practical limitations of malware analysis procedures. We here show how these aspects are correlated with previous work's reports. For reader's convenience, the considered papers are summarized in Table 5.

Table 5: **Related Work.** Summary of differences.

Science of Security		
Work	**Approach**	**Issues**
Ours	Practical	Experiment design
[82]	Theoretical	Results reporting
Security Limits		
Work	**Approach**	**Issues**
Ours	Practical	AV labels, private datasets
[34]	Theoretical	Path explosition
[139]	Theoretical	Opaque Constants
Pitfalls		
Work	**Approach**	**Issues**
Ours	Practical	Signatures, datasets
[9]	Theoretical	False Positives
[153]	Theoretical	Training data
Sandbox		
Work	**Approach**	**Issues**
Ours	Practical	sinkholing, loading
[194]	Practical	evasion
[127]	Practical	fingerprint
[168]	Practical	stimulation
[105]	Practical	replay

Science of Security. Discussion about computer viruses myths dates back to the 1990's [164], but the development of solutions have taken the forefront of the field at the expense of in-depth scientific discussions. In this work, we revisited in-depth discussions on the field by systematizing pitfalls and challenges in malware research. We highlighted that all types of malware research (engineering solutions, offensive research, observational studies, and network traffic) can be conducted according to a method integrating the scientific and engineering methods. Herley and van Oorschot recently discussed the elusive goal of security as a scientific pursuit [82], and identified reproducibility issues in current research papers. In this work, we complement this discussion in-depth and in the context of malware research with issues that go beyond reproducibility.
Prior work investigating malware research challenges fit in one of the following categories:

Theoretical Limitations. Prior work investigated the limits and constraints of malware analysis procedures, which can appear naturally or be actively exploited by attackers to make sample detection harder. Typically,

these constructions consist of ambiguous execution and data flows, which are hard to be tracked because they span an exponential number of paths [34]. In addition, constructions such as opaque constants [139] cannot be statically solved, thus requiring runtime analysis, which raises processing costs and demands more time, limiting scale. Understanding these limitations is important to properly define threat models and establish clear research review guidelines. In this work, we complemented this prior literature by extending the analysis of theoretical limits of malware experiments to also include practical considerations.

Experiment Design Issues. As important as to understand the limits of data collection procedures is to understand the limits of analysis procedures, which affect the experiment design. A poorly designed experiment may result in the reporting of results that are not reproducible or applicable in actual scenarios. Axelsson has already reported issues with experiment design, as in the "base-rate fallacy for IDS" [9], which states that "*the factor limiting the performance of an intrusion detection system is not the ability to identify behavior correctly as intrusive, but rather its ability to suppress false alarms*". In other words, a solution reporting too many FP is impractical for actual scenarios, despite presenting high TP (True Positive) detection rates.

A large number of current malware research rely machine learning methods. Therefore, similar to the base-rate fallacy for IDS, Pendlebury et al. [153] also reported multiple bias while training models for security solutions, such as datasets not reflecting realistic conditions [153]. Unfortunately, unrealistic datasets and threat models are often seem in malware research. This paper extended this discussion to malware experiments in general and discussed their impact in the development of the malware research as a methodologically strong discipline.

Sandbox Issues. One of the most frequent concerns researcher have when developing malware experiments is regarding the sandbox environment leveraged for performing real-time sample analysis, given the multiple challenges that the use of this type of solution imposes. Previous work on the literature have already identified some challenges with using sandboxes in experiments, for example, sandbox evasion [194] due to fingerprinting [127] or lack of proper stimulation [168]. Kirat et al. [105] also highlighted the need for isolating and replaying network packets for proper sample execution across distinct sandboxes. Given these challenges, malware research often fails in accomplishing some of the analysis requirements, as discussed by Rossow et al. [166]. In this work, we presented additional aspects, such as identifying broken samples execution and the lack of malware loaders, which must be considered in the development of malware research work.

Improving Research Practices. In this work, we proposed that all malware research can be done via a methodology that integrates the scientific and the engineering methods. Fortunately, this need has been acknowledged (yet slowly and unstructuredly) by the community in recent years, via guidelines for handling domain lists [167], generating dataset for static analysis procedures [131], for benchmarking systems [190], and for the application of machine learning [176, 8, 178, 68, 37]. We hope our work to motivate other researchers towards developing best practices guidelines based on the lessons we learned and recommendations provided.

New views of security. A major contribution of this work is to position security among the scientific and the engineering methods. Whereas we believe this might be a significant advance, these are not the only factors to be considered in a security analysis. For instance, we believe that economic aspects of security [5] should also be considered in analyses procedures. Thus, we expected that these might be incorporated in future research methodologies.

7 Conclusion

In this paper, we presented a systematic literature review of scholarly work in the field of malware analysis and detection published in the major security conferences in the period between 2000 and 2018. Our analysis encompassed a total of 491 papers, which, to the best of our knowledge, is the largest literature review of its kind presented so-far. Moreover, unlike previous research work, our analysis is not limited to surveying the distinct kinds of published work, but we also delve into their methodological approaches and experimental design practices to identify the challenges and the pitfalls of malware research. We identified a set of 20 pitfalls and challenges commonly related to malware research, that range from the lack of a proper threat model definition to the adoption of closed-source solutions and private datasets that do not streamline reproducibility. To help overcoming these challenges and avoiding the pitfalls, we proposed a set of actionable items to be considered by the malware research community: i) Consolidating malware research as a diversified research field with different needed types of research (e.g., engineering solutions, offensive research, observational/landscape studies, and network traffic); (ii) design of engineering solutions with clearer, direct assumptions (e.g., positioning solutions as proofs-of-concept vs. deployable, offline vs. online, etc.); iii) Design of experiments to reflecting more realistic scenarios instead of generalized cases the scenario where solution should operate (e.g., corporation, home, lab, country)leveraging datasets having malware families balanced to reflect specific countries, vulnerable populations or corporations); and iv) Acknowledgment of limitations

current technologies and norms exert in research existing solutions limitations (e.g., the use of closed-source AV solutions as groundtruth for malware experiments) to support the development of more reproducible research. We hope that our insights might help fostering, particularly, the next-generation of anti-malware solutions and, more broadly, the malware research field as a more mature scientific field.

We reinforce once again that the views presented in this work are not unique; other interpretations of the observed phenomenon are possible. In these cases, the researchers must formalize their views so as we can build a body of knowledge on methodological practices. This type of body of knowledge might be a recommended reading for students entering the field and might also work as a basis for the development of future guidelines.

Acknowledgements. Marcus thanks the Brazilian National Counsel of Technological and Scientific Development (CNPq) for the PhD Scholarship 164745/2017-3. Daniela on behalf of all authors thanks the National Science Foundation (NSF) by the project grant CNS-1552059.

References

[1] ACM. Computing surveys. `https://csur.acm.org/`, 2019.

[2] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 76–82. IEEE, 2018.

[3] Joey Allen, Matthew Landen, Sanya Chaba, Yang Ji, Simon Pak Ho Chung, and Wenke Lee. Improving accuracy of android malware detection with lightweight contextual awareness. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, page 210–221, New York, NY, USA, 2018. Association for Computing Machinery.

[4] Sumayah Alrwais, Kan Yuan, Eihal Alowaisheq, Xiaojing Liao, Alina Oprea, XiaoFeng Wang, and Zhou Li. Catching predators at watering holes: Finding and understanding strategically compromised websites. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ACSAC '16, pages 153–166. ACM, 2016.

[5] Ross Anderson and Tyler Moore. The economics of information security. `https://www.cl.cam.ac.uk/~rja14/Papers/sciecon2.pdf`, 2005.

[6] Dennis Andriesse and Herbert Bos. Instruction-level steganography for covert trigger-based malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 41–50. Springer, 2014.

[7] Ionut Arghire. Windows 7 most hit by wannacry ransomware. `http://www.securityweek.com/windows-7-most-hit-wannacry-ransomware`, 2017.

[8] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security, 2020.

[9] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, August 2000.

[10] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The nepenthes platform: An efficient approach to collect malware. In Diego Zamboni and Christopher Kruegel, editors, *Recent Advances in Intrusion Detection*, pages 165–184, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[11] Davide Balzarotti. System security circus. `http://s3.eurecom.fr/~balzarot/notes/top4_2018/`, 2018.

[12] Sebastian Banescu, Christian Collberg, Vijay Ganesh, Zack Newsham, and Alexander Pretschner. Code obfuscation against symbolic execution attacks. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 189–200. ACM, 2016.

[13] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. A view on current malware behaviors. In *Proceedings of the 2Nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, LEET'09, pages 8–8, Berkeley, CA, USA, 2009. USENIX Association.

[14] Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda. Dynamic analysis of malicious code. *Journal in Computer Virology*, 2(1):67–77, 2006.

[15] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. dr0wned–cyber-physical attack with additive manufacturing. In *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[16] Leyla Bilge and Tudor Dumitraş. Before we knew it: An empirical study of zero-day attacks in the

real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 833–844, New York, NY, USA, 2012. ACM.

[17] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur.*, 16(4):14:1–14:28, April 2014.

[18] BitDefender. The update system for virus signatures. `https://www.bitdefender.com/support/the-update-system-for-virus-signatures-216.html`.

[19] Thomas Bläsing, Leonid Batyuk, Aubrey-Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak. An android application sandbox system for suspicious software detection. In *2010 5th International Conference on Malicious and Unwanted Software*, pages 55–62. IEEE, 2010.

[20] Stevens Le Blond, Cedric Gilbert, Utkarsh Upadhyay, Manuel Gomez Rodriguez, and David Choffnes. A broad view of the ecosystem of socially engineered exploit documents. `https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/broad-view-ecosystem-socially-engineered-exploit-documents/`, 2017.

[21] Lorenzo Bordoni, Mauro Conti, and Riccardo Spolaor. Mirage: Toward a stealthier and modular malware analysis sandbox for android. In *European Symposium on Research in Computer Security*, pages 278–296. Springer, 2017.

[22] Marcus Botacin, Giovanni Bertão, Paulo de Geus, André Grégio, Christopher Kruegel, and Giovanni Vigna. On the security of application installers and online software repositories. In Clémentine Maurice, Leyla Bilge, Gianluca Stringhini, and Nuno Neves, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 192–214, Cham, 2020. Springer International Publishing.

[23] Marcus Botacin, Paulo Lício de Geus, and André Grégio. "vanilla" malware: vanishing antiviruses by interleaving layers and layers of attacks. *Journal of Computer Virology and Hacking Techniques*, Jun 2019.

[24] Marcus Botacin, Paulo Lício De Geus, and André grégio. Who watches the watchmen: A security-focused review on current state-of-the-art techniques, tools, and methods for systems and binary analysis on modern platforms. *ACM Comput. Surv.*, 51(4):69:1–69:34, July 2018.

[25] Marcus Felipe Botacin, Paulo Lício de Geus, and André Ricardo Abed Grégio. The other guys: automated analysis of marginalized malware. *Journal of Computer Virology and Hacking Techniques*, 14(1):87–98, Feb 2018.

[26] Michael Brengel and Christian Rossow. M em s crimper: Time-and space-efficient storage of malware sandbox memory dumps. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 24–45. Springer, 2018.

[27] Matthew Brocker and Stephen Checkoway. iseeyou: Disabling the macbook webcam indicator LED. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 337–352, San Diego, CA, 2014. USENIX Association.

[28] Erik Buchanan, Ryan Roemer, Hovav Shacham, and Stefan Savage. When good instructions go bad: Generalizing return-oriented programming to risc. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 27–38. ACM, 2008.

[29] Alejandro Calleja, Juan Tapiador, and Juan Caballero. A look into 30 years of malware development from a software metrics perspective. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 325–345. Springer, 2016.

[30] D. Carlin, A. Cowan, P. O'Kane, and S. Sezer. The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes. *IEEE Access*, 5:17742–17752, 2017.

[31] Nicholas Carlini, Antonio Barresi, Mathias Payer, David Wagner, and Thomas R. Gross. Control-flow bending: On the effectiveness of control-flow integrity. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 161–176, Washington, D.C., 2015. USENIX Association.

[32] Nicholas Carlini and David Wagner. {ROP} is still dangerous: Breaking modern defenses. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 385–399, 2014.

[33] Nicholas Carlini and David Wagner. ROP is still dangerous: Breaking modern defenses. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 385–399, San Diego, CA, 2014. USENIX Association.

[34] Lorenzo Cavallaro, Prateek Saxena, and R. Sekar. On the limits of information flow techniques for

malware analysis and containment. In Diego Zamboni, editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 143–163, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[35] F. Ceschin, F. Pinage, M. Castilho, D. Menotti, L. S. Oliveira, and A. Gregio. The need for speed: An analysis of brazilian malware classifers. *IEEE Security & Privacy*, 16(6):31–41, Nov.-Dec. 2018.

[36] Fabrício Ceschin, Marcus Botacin, Heitor Murilo Gomes, Luiz S. Oliveira, and André Grégio. Shallow security: On the creation of adversarial variants to evade machine learning-based malware detectors. In *Proceedings of the 3rd Reversing and Offensive-Oriented Trends Symposium*, ROOTS'19, New York, NY, USA, 2019. Association for Computing Machinery.

[37] Fabrício Ceschin, Heitor Murilo Gomes, Marcus Botacin, Albert Bifet, Bernhard Pfahringer, Luiz S. Oliveira, and André Grégio. Machine learning (in) security: A stream of problems, 2020.

[38] Orçun Çetin, Carlos Gañán, Lisette Altena, Samaneh Tajalizadehkhoob, and Michel van Eeten. Let me out! evaluating the effectiveness of quarantining compromised users in walled gardens. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 251–263, Baltimore, MD, 2018. USENIX Association.

[39] S. K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D. G. Andersen. Splitscreen: Enabling efficient, distributed malware detection. *Journal of Communications and Networks*, 13(2):187–200, 2011.

[40] Daming D Chen, Maverick Woo, David Brumley, and Manuel Egele. Towards automated dynamic analysis for linux-based embedded firmware. In *NDSS*, pages 1–16, 2016.

[41] Haining Chen, Ninghui Li, William Enck, Yousra Aafer, and Xiangyu Zhang. Analysis of seandroid policies: Combining mac and dac in android. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, pages 553–565, New York, NY, USA, 2017. ACM.

[42] Lingwei Chen, Shifu Hou, and Yanfang Ye. Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, pages 362–372, New York, NY, USA, 2017. ACM.

[43] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In *2017 European Intelligence and Security Informatics Conference (EISIC)*, pages 99–106. IEEE, 2017.

[44] Ping Chen, Hai Xiao, Xiaobin Shen, Xinchun Yin, Bing Mao, and Li Xie. Drop: Detecting return-oriented programming malicious code. In *International Conference on Information Systems Security*, pages 163–177. Springer, 2009.

[45] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *computers & security*, 73:326–344, 2018.

[46] Binlin Cheng, Jiang Ming, Jianmin Fu, Guojun Peng, Ting Chen, Xiaosong Zhang, and Jean-Yves Marion. Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 395–411, New York, NY, USA, 2018. ACM.

[47] Yueqiang Cheng, Zongwei Zhou, Yu Miao, Xuhua Ding, and Huijie Robert Deng. Ropecker: A generic and practical approach for defending against rop attacks. *In Symposium on Network and Distributed System Security (NDSS*, 2014.

[48] Cisco. Updating anti-virus signatures. `https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/af1321261.html`.

[49] ClamAV. Clamavnet. `https://www.clamav.net/`, 2019.

[50] ClamTk. Updating antivirus signatures. `http://clamtk.sourceforge.net/help/update-signatures-clamtk.html`.

[51] Michele Colajanni, Daniele Gozzi, and Mirco Marchetti. Collaborative architecture for malware detection and analysis. In *IFIP International Information Security Conference*, pages 79–93. Springer, 2008.

[52] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, January 1989.

[53] Lucian Constantin. Researcher wins $200,000 prize from microsoft for new exploit mitigation technology. `https://www.pcworld.com/article/`

259943/researcher_wins_200000_prize_from_microsoft_for_new_exploit_mitigation_technology.html, 2012.

[54] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. Understanding linux malware. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 161–175. IEEE, 2018.

[55] Weidong Cui, Marcus Peinado, Zhilei Xu, and Ellick Chan. Tracking rootkit footprints with a practical memory analysis system. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 601–615, 2012.

[56] Weidong Cui, Marcus Peinado, Zhilei Xu, and Ellick Chan. Tracking rootkit footprints with a practical memory analysis system. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 601–615, Bellevue, WA, 2012. USENIX.

[57] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3422–3426. IEEE, 2013.

[58] Peter J. Denning. The science in computer science. *Commun. ACM*, 56(5):35–38, May 2013.

[59] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. Keep me updated: An empirical study of third-party library updatability on android. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2187–2200. ACM, 2017.

[60] Jean-Marc Desharnais and Alain April. Software maintenance productivity and maturity. In *Proceedings of the 11th International Conference on Product Focused Software*, PROFES '10, page 121–125, New York, NY, USA, 2010. Association for Computing Machinery.

[61] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[62] Yue Duan, Mu Zhang, Abhishek Vasisht Bhaskar, Heng Yin, Xiaorui Pan, Tongxin Li, Xueqiang Wang, and X Wang. Things you may not know about android (un) packers: a systematic study based on whole-system emulation. In *25th Annual Network and Distributed System Security Symposium, NDSS*, pages 18–21, 2018.

[63] Duo. Security report finds phishing, not zero-days, is the top malware infection vector. https://duo.com/blog/security-report-finds-phishing-not-zero-days-is-the-top-malware-infection-vector, 2018.

[64] Nicholas Epley and Thomas Gilovich. The anchoring-and-adjustment heuristic: Why the adjustments are insufficient. *Psychological Science*, 17(4):311–318, 2006. PMID: 16623688.

[65] Qian Feng, Aravind Prakash, Heng Yin, and Zhiqiang Lin. Mace: High-coverage and robust memory analysis for commodity operating systems. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 196–205. ACM, 2014.

[66] Yu Feng, Osbert Bastani, Ruben Martins, Isil Dillig, and Saswat Anand. Automated synthesis of semantic malware signatures using maximum satisfiability. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/automated-synthesis-semantic-malware-signatures-using-maximum-satisfiability/, 2017.

[67] Jacob Fustos, Farzad Farshchi, and Heechul Yun. Spectreguard: An efficient data-centric defense mechanism against spectre attacks. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 61:1–61:6, New York, NY, USA, 2019. ACM.

[68] Giorgio Giacinto and Belur V. Dasarathy. An editorial note to prospective authors: Machine learning for computer security: A guide to prospective authors. *Inf. Fusion*, 12(3):238–239, July 2011.

[69] Jason Gionta, Ahmed Azab, William Enck, Peng Ning, and Xiaolan Zhang. Seer: Practical memory virus scanning as a service. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, page 186–195, New York, NY, USA, 2014. Association for Computing Machinery.

[70] Jan Goebel, Thorsten Holz, and Carsten Willems. Measurement and analysis of autonomous spreading malware in a university environment. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 109–128. Springer, 2007.

[71] Enes Göktaş, Elias Athanasopoulos, Michalis Polychronakis, Herbert Bos, and Georgios Portokalidis. Size does matter: Why using gadget-chain length to prevent code-reuse attacks is hard. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 417–432, San Diego, CA, 2014. USENIX Association.

[72] Liangyi Gong, Zhenhua Li, Feng Qian, Zifan Zhang, Qi Alfred Chen, Zhiyun Qian, Hao Lin, and Yunhao Liu. Experiences of landing machine learning onto market-scale mobile malware detection. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery.

[73] B. Govindarajalu. *Computer Architecture and Organization: Design Principles and Applications 2nd Edition*. Mc Graw Hill India, 2017.

[74] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 281–294. ACM, 2012.

[75] Mariano Graziano, Davide Canali, Leyla Bilge, Andrea Lanzi, Elaine Shi, Davide Balzarotti, Marten van Dijk, Michael Bailey, Srinivas Devadas, Mingyan Liu, et al. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 1057–1072, 2015.

[76] Mariano Graziano, Corrado Leita, and Davide Balzarotti. Towards network containment in malware analysis systems. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 339–348. ACM, 2012.

[77] André Ricardo Abed Grégio, Paulo Lício De Geus, Christopher Kruegel, and Giovanni Vigna. Tracking memory writes for malware classification and code reuse identification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 134–143. Springer, 2012.

[78] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[79] André Ricardo Abed Grégio, Vitor Monte Afonso, Dario Simões Fernandes Filho, Paulo Lício de Geus, and Mario Jino. Toward a Taxonomy of Malware Behaviors. *The Computer Journal*, 58(10):2758–2777, 07 2015.

[80] Guofei Gu, Phillip Porras, Vinod Yegneswaran, and Martin Fong. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *16th USENIX Security Symposium (USENIX Security 07)*, Boston, MA, 2007. USENIX Association.

[81] Mordechai Guri and Dima Bykhovsky. air-jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (ir). *Computers & Security*, 82:15–29, 2019.

[82] C. Herley and P. C. v. Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 99–120, May 2017.

[83] Chi-Yao Hong, Fang Yu, and Yinglian Xie. Populated ip addresses: Classification and applications. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 329–340. ACM, 2012.

[84] Francis Hsu, Hao Chen, Thomas Ristenpart, Jason Li, and Zhendong Su. Back to the future: A framework for automatic malware removal and system repair. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 257–268. IEEE, 2006.

[85] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631, May 2018.

[86] Wenyi Huang and Jack W. Stokes. Mtnet: A multi-task neural network for dynamic malware classification. In Juan Caballero, Urko Zurutuza, and Ricardo J. Rodríguez, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 399–418, Cham, 2016. Springer International Publishing.

[87] IACR. Real world crypto symposium. `https://rwc.iacr.org/`, 2019.

[88] Daisuke Inoue, Masashi Eto, Katsunari Yoshioka, Shunsuke Baba, Kazuya Suzuki, Junji Nakazato, Kazuhiro Ohtaka, and Koji Nakao. nicter: An incident analysis system toward binding network monitoring with malware analysis. In *2008 WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, pages 58–66. IEEE, 2008.

[89] Daisuke Inoue, Katsunari Yoshioka, Masashi Eto, Yuji Hoshizawa, and Koji Nakao. Malware behavior analysis in isolated miniature network for revealing malware's network activity. In *2008 IEEE International Conference on Communications*, pages 1715–1721. IEEE, 2008.

[90] Kyriakos K. Ispoglou and Mathias Payer. malwash: Washing malware to evade dynamic analysis. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, 2016. USENIX Association.

[91] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 19–35, 2018.

[92] Suman Jana, Donald E Porter, and Vitaly Shmatikov. Txbox: Building secure, efficient sandboxes with system transactions. In *2011 IEEE Symposium on Security and Privacy*, pages 329–344. IEEE, 2011.

[93] Yeongjin Jang, Chengyu Song, Simon P Chung, Tielei Wang, and Wenke Lee. A11y attacks: Exploiting accessibility in operating systems. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–115. ACM, 2014.

[94] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 128–138. ACM, 2007.

[95] Roberto Jordaney, Kumar Sharad, Santanu K. Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 625–642, Vancouver, BC, 2017. USENIX Association.

[96] Murat Kantarcioglu and Bowei Xi. Adversarial data mining: Big data meets cyber security. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1866–1867. ACM, 2016.

[97] Nikos Karampatziakis, Jack W Stokes, Anil Thomas, and Mady Marinescu. Using file relationships in malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 1–20. Springer, 2012.

[98] Evanson Mwangi Karanja, Shedden Masupe, and Mandu Gasennelwe-Jeffrey. Challenge paper: Towards open datasets for internet of things malware. *J. Data and Information Quality*, 10(2):7:1–7:5, September 2018.

[99] Khaled N. Khasawneh, Meltem Ozsoy, Caleb Donovick, Nael Abu-Ghazaleh, and Dmitry Ponomarev. Ensemble learning for low-level hardware-supported malware detection. In Herbert Bos, Fabian Monrose, and Gregory Blanc, editors, *Research in Attacks, Intrusions, and Defenses*, pages 3–25, Cham, 2015. Springer International Publishing.

[100] Yosuke Kikuchi, Hiroshi Mori, Hiroki Nakano, Katsunari Yoshioka, Tsutomu Matsumoto, and Michel Van Eeten. Evaluating malware mitigation by android market operators. In *9th Workshop on Cyber Security Experimentation and Test ({CSET} 16)*, 2016.

[101] Doowon Kim, Bum Jun Kwon, and Tudor Dumitraş. Certified malware: Measuring breaches of trust in the windows code-signing pki. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1435–1448, New York, NY, USA, 2017. ACM.

[102] Jin-Young Kim, Seok-Jun Bu, and Sung-Bae Cho. Malware detection using deep transferred generative adversarial networks. In *International Conference on Neural Information Processing*, pages 556–564. Springer, 2017.

[103] Johannes Kinder, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. Detecting malicious code by model checking. In Klaus Julisch and Christopher Kruegel, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 174–187, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[104] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barebox: Efficient malware analysis on bare-metal. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 403–412, New York, NY, USA, 2011. ACM.

[105] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barecloud: Bare-metal analysis-based evasive malware detection. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 287–301, San Diego, CA, 2014. USENIX Association.

[106] Bojan Kolosnjaji, Apostolis Zarras, Tamas Lengyel, George Webster, and Claudia Eckert. Adaptive semantics-aware malware classification. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721*, DIMVA 2016, pages 419–439, Berlin, Heidelberg, 2016. Springer-Verlag.

[107] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence*, pages 137–149. Springer, 2016.

[108] David Korczynski and Heng Yin. Capturing malware propagations with code injections and code-reuse attacks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1691–1708. ACM, 2017.

[109] Christoforos E. Kozyrakis and David A. Patterson. A new direction for computer architecture research. `https://web.stanford.edu/~kozyraki/publications/1998.IEEEComputer.Direction.pdf`, 1998.

[110] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[111] Bum Jun Kwon, Jayanta Mondal, Jiyong Jang, Leyla Bilge, and Tudor Dumitraş. The dropper effect: Insights into malware distribution with downloader graph analytics. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1118–1129, New York, NY, USA, 2015. ACM.

[112] Alexander Küchler, Alessandro Mantovani, Yufei Han, Leyla Bilge, and Davide Balzarotti. Does every second count?time-based evolution of malware behavior in sandboxes. `http://s3.eurecom.fr/docs/ndss21_kuechler.pdf`, 2021.

[113] Fanny Lalonde Levesque, Jude Nsiempba, José M. Fernandez, Sonia Chiasson, and Anil Somayaji. A clinical study of risk factors related to malware infections. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13. ACM, 2013.

[114] Andrea Lanzi, Davide Balzarotti, Christopher Kruegel, Mihai Christodorescu, and Engin Kirda. Accessminer: using system-centric models for malware protection. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 399–412. ACM, 2010.

[115] Pavel Laskov and Richard Lippmann. Machine learning in adversarial environments, 2010.

[116] Alan Lee, Vijay Varadharajan, and Udaya Tupakula. On malware characterization and attack classification. In *Proceedings of the First Australasian Web Conference - Volume 144*, AWC '13, page 43–47, AUS, 2013. Australian Computer Society, Inc.

[117] Johnny Chung Lee. Hacking the nintendo wii remote. *IEEE pervasive computing*, 7(3):39–45, 2008.

[118] Corrado Leita and Marc Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *2008 Seventh European Dependable Computing Conference*, pages 99–109. IEEE, 2008.

[119] Corrado Leita, Marc Dacier, and Frederic Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with scriptgen based honeypots. In *International Workshop on Recent Advances in Intrusion Detection*, pages 185–205. Springer, 2006.

[120] Chaz Lever, Platon Kotzias, Davide Balzarotti, Juan Caballero, and Manos Antonakakis. A lustrum of malware network communication: Evolution and insights. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 788–804. IEEE, 2017.

[121] Fanny Lalonde Levesque and Jose M. Fernandez. Computer security clinical trials: Lessons learned from a 4-month pilot study. In *7th Workshop on Cyber Security Experimentation and Test (CSET 14)*, San Diego, CA, 2014. USENIX Association.

[122] Xinghua Li, Jianfeng Ma, and SangJae Moon. On the security of the canetti-krawczyk model. *International Conference on Computational and Information Science*, 2005.

[123] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao, and Brian Chavez. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[124] Zhiqiang Lin, Xiangyu Zhang, and Dongyan Xu. Automatic reverse engineering of data structures from binary execution. In *Proceedings of the 11th Annual Information Security Symposium*, CERIAS '10, pages 5:1–5:1, West Lafayette, IN, 2010. CERIAS - Purdue University.

[125] Martina Lindorfer, Alessandro Di Federico, Federico Maggi, Paolo Milani Comparetti, and Stefano Zanero. Lines of malicious code: Insights into the malicious software industry. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 349–358. ACM, 2012.

[126] Martina Lindorfer, Matthias Neugschwandtner, Lukas Weichselbaum, Yanick Fratantonio, Victor van der Veen, and Christian Platzer. Andrubis – 1,000,000 apps later: A view on current android malware behaviors. In *Proceedings of the 2014*

Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS '14, pages 3–17, Washington, DC, USA, 2014. IEEE Computer Society.

[127] Ke Liu, Shuai Lu, and Chaoge Liu. Poster: Fingerprinting the publicly available sandboxes. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1469–1471, New York, NY, USA, 2014. ACM.

[128] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, pages 190–200, New York, NY, USA, 2005. ACM.

[129] Meng Luo, Oleksii Starov, Nima Honarmand, and Nick Nikiforakis. Hindsight: Understanding the evolution of ui vulnerabilities in mobile browsers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 149–162, New York, NY, USA, 2017. ACM.

[130] Weiqin Ma, Pu Duan, Sanmin Liu, Guofei Gu, and Jyh-Charn Liu. Shadow attacks: Automatically evading system-call-behavior based malware detection. *J. Comput. Virol.*, 8(1-2):1–13, May 2012.

[131] Aravind Machiry, Nilo Redini, Eric Gustafson, Hojjat Aghakhani, Christopher Kruegel, and Giovanni Vigna. Towards automatically generating a sound and complete dataset for evaluating static analysis tools. `https://ruoyuwang.me/bar2019/pdfs/bar2019-final90.pdf`, 2019.

[132] Giorgi Maisuradze, Michael Backes, and Christian Rossow. What cannot be read, cannot be leveraged? revisiting assumptions of jit-rop defenses. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 139–156, Austin, TX, 2016. USENIX Association.

[133] MalwareBytes. Explained yara rules. `https://blog.malwarebytes.com/security-world/technology/2017/09/explained-yara-rules/`, 2017.

[134] Pratyusa K Manadhata, Sandeep Yadav, Prasad Rao, and William Horne. Detecting malicious domains via graph inference. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2014.

[135] Microsoft. Review event logs and error codes to troubleshoot issues with microsoft defender antivirus. `https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-antivirus/troubleshoot-microsoft-defender-antivirus`, 2018.

[136] Najmeh Miramirkhani, Mahathi Priya Appini, Nick Nikiforakis, and Michalis Polychronakis. Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 1009–1024. IEEE, 2017.

[137] Shinsuke Miwa, Toshiyuki Miyachi, Masashi Eto, Masashi Yoshizumi, and Yoichi Shinoda. Design and implementation of an isolated sandbox with mimetic internet used to analyze malwares. In *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, DETER, pages 6–6, Berkeley, CA, USA, 2007. USENIX Association.

[138] Tyler Moore, Nektarios Leontiadis, and Nicolas Christin. Fashion crimes: Trending-term exploitation on the web. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 455–466. ACM, 2011.

[139] Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of static analysis for malware detection. In *Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual*, pages 421–430. IEEE, 2007.

[140] Alexander Moshchuk, Tanya Bragin, Steven D Gribble, and Henry M Levy. A crawler-based study of spyware in the web. In *NDSS*, volume 1, page 2, 2006.

[141] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Understanding the prevalence and use of alternative plans in malware with network games. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 1–10, New York, NY, USA, 2011. ACM.

[142] Antonio Nappa, M Zubair Rafique, and Juan Caballero. Driving in the cloud: An analysis of drive-by download operations and abuse reporting. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 1–20. Springer, 2013.

[143] NASA. Mission, goals, objectives. `https://www.nasa.gov/offices/emd/home/mgo.html`, 2019.

[144] NASA. Nasa cost estimating handbook (ceh). https://www.nasa.gov/offices/ocfo/nasa-cost-estimating-handbook-ceh, 2019.

[145] NDSS. Laser workshop. https://www.ndss-symposium.org/ndss2021/laser-workshop-2021, 2021.

[146] Matthias Neugschwandtner, Paolo Milani Comparetti, Gregoire Jacob, and Christopher Kruegel. Forecast: skimming off the malware cream. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 11–20. ACM, 2011.

[147] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. Made: Security analytics for enterprise threat detection. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, pages 124–136, New York, NY, USA, 2018. ACM.

[148] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56. IEEE, 2015.

[149] Gerhard Pahl and Wolfgang Beitz. *Engineering design: a systematic approach*. Springer Science & Business Media, 2013.

[150] Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis. Transparent ROP exploit mitigation using indirect branch tracing. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 447–462, Washington, D.C., 2013. USENIX.

[151] Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1916–1920. IEEE, 2015.

[152] Paul Pearce, Vacha Dave, Chris Grier, Kirill Levchenko, Saikat Guha, Damon McCoy, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. Characterizing large-scale click fraud in zeroaccess. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 141–152, New York, NY, USA, 2014. ACM.

[153] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. Enabling fair ml evaluations for security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 2264–2266, New York, NY, USA, 2018. ACM.

[154] Fei Peng, Zhui Deng, Xiangyu Zhang, Dongyan Xu, Zhiqiang Lin, and Zhendong Su. X-force: Force-executing binary programs for security applications. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 829–844, San Diego, CA, 2014. USENIX Association.

[155] Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In *2008 Annual Computer Security Applications Conference (ACSAC)*, pages 301–310. IEEE, 2008.

[156] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *NSDI*, volume 10, page 14, 2010.

[157] Karl Popper. *The logic of scientific discovery*. Routledge, 1959. Republished: 2005.

[158] Rebecca S Portnoff, Linda N Lee, Serge Egelman, Pratyush Mishra, Derek Leung, and David Wagner. Somebody's watching me?: Assessing the effectiveness of webcam indicator lights. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1649–1658. ACM, 2015.

[159] Prisma. Transparent reporting of systematic reviews and meta-analyses. http://www.prisma-statement.org/, 2019.

[160] Zhiyun Qian, Z Morley Mao, and Yinglian Xie. Collaborative tcp sequence number inference attack: how to crack sequence number under a second. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 593–604. ACM, 2012.

[161] M. Zubair Rafique and Juan Caballero. Firma: Malware clustering and network signature generation with mixed network behaviors. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions, and Defenses - Volume 8145*, RAID 2013, pages 144–163, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[162] M Zubair Rafique and Juan Caballero. Firma: Malware clustering and network signature generation with mixed network behaviors. In *International Workshop on Recent Advances in Intrusion Detection*, pages 144–163. Springer, 2013.

[163] Mohd Faizal Ab Razak, Nor Badrul Anuar, Rosli Salleh, and Ahmad Firdaus. The rise of "malware":

Bibliometric analysis of malware study. *Journal of Network and Computer Applications*, 75:58 – 76, 2016.

[164] Rob Rosenberger and Ross Greenberg. Computer virus myths. *SIGSAC Rev.*, 7(4):21–24, January 1990.

[165] Christian Rossow, Christian Dietrich, and Herbert Bos. Large-scale analysis of malware downloaders. In *Proceedings of the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'12, pages 42–61, Berlin, Heidelberg, 2013. Springer-Verlag.

[166] Christian Rossow, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten van Steen. Prudent practices for designing malware experiments: Status quo and outlook. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 65–79, Washington, DC, USA, 2012. IEEE Computer Society.

[167] Walter Rweyemamu, Tobias Lauinger, Christo Wilson, William Robertson, and Engin Kirda. Clustering and the weekend effect: Recommendations for the use of top domain lists in security research. In David Choffnes and Marinho Barcellos, editors, *Passive and Active Measurement*, pages 161–177, Cham, 2019. Springer International Publishing.

[168] Aleieldin Salem. Stimulation and detection of android repackaged malware with active learning. `https://arxiv.org/pdf/1808.01186.pdf`, 2018.

[169] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. Phoenix: Dga-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 192–211. Springer, 2014.

[170] James Scott. Signature based malware detection is dead. `https://pdfs.semanticscholar.org/646c/8b08dd5c3c70785550eab01e766798be80b5.pdf`, 2017.

[171] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In Fabian Monrose, Marc Dacier, Gregory Blanc, and Joaquin Garcia-Alfaro, editors, *Research in Attacks, Intrusions, and Defenses*, pages 230–253, Cham, 2016. Springer International Publishing.

[172] Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig. Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 335–350, New York, NY, USA, 2007. ACM.

[173] M. Zubair Shafiq, Syed Ali Khayam, and Muddassar Farooq. Embedded malware detection using markov n-grams. In Diego Zamboni, editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 88–107, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[174] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[175] Andrew Slaughter, Mark Yampolskiy, Manyalibo Matthews, Wayne E King, Gabe Guss, and Yuval Elovici. How to ensure bad quality in metal additive manufacturing: In-situ infrared thermography from the security perspective. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 78. ACM, 2017.

[176] Michael R. Smith, Nicholas T. Johnson, Joe B. Ingram, Armida J. Carbajal, Bridget I. Haus, Eva Domschot, Ramyaa Ramyaa, Christopher C. Lamb, Stephen J. Verzi, and W. Philip Kegelmeyer. Mind the gap: On bridging the semantic gap between machine learning and malware analysis. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, AISec'20, page 49–60, New York, NY, USA, 2020. Association for Computing Machinery.

[177] Tomas Sochor and Matej Zuzcak. Study of internet threats and attack methods using honeypots and honeynets. In *International Conference on Computer Networks*, pages 118–127. Springer, 2014.

[178] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, 2010.

[179] IEEE S&P. Ieee security & privacy. `https://www.ieee-security.org/TC/SP2020/cfpapers.html`, 2019.

[180] Blaine Stancill, Kevin Z Snow, Nathan Otterness, Fabian Monrose, Lucas Davi, and Ahmad-Reza Sadeghi. Check my profile: Leveraging static analysis for fast and accurate detection of rop gadgets. In *International Workshop on Recent Advances in Intrusion Detection*, pages 62–81. Springer, 2013.

[181] Jay Stokes, , , Joe Faulhaber, Mady Marinescu, Anil Thomas, and Marius Gheorghescu. Scalable telemetry classification for automated malware detection. In *Proceedings of European Symposium on Research in Computer Security (ESORICS2012)*. Springer, September 2012.

[182] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 635–647, New York, NY, USA, 2009. ACM.

[183] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 133–144, New York, NY, USA, 2013. ACM.

[184] Nina Sunde and Itiel E. Dror. Cognitive and human factors in digital forensics: Problems, challenges, and the way forward. *Digital Investigation*, 29:101 – 108, 2019.

[185] Vasilis G. Tasiopoulos and Sokratis K. Katsikas. Bypassing antivirus detection with encryption. In *Proceedings of the 18th Panhellenic Conference on Informatics*, PCI '14, pages 16:1–16:2, New York, NY, USA, 2014. ACM.

[186] BIML Team. Annotated bibliography. `https://berryvilleiml.com/references/`, 2020.

[187] Xabier Ugarte-Pedrero, Davide Balzarotti, Igor Santos, and Pablo G Bringas. Sok: Deep packer inspection: A longitudinal study of the complexity of run-time packers. In *2015 IEEE Symposium on Security and Privacy*, pages 659–673. IEEE, 2015.

[188] USENIX. Usenix soups. `https://www.usenix.org/conference/soups2019`, 2019.

[189] USENIX. Workshop on cyber security experimentation and test. `https://www.usenix.org/conferences/byname/135`, 2020.

[190] Erik van der Kouwe, Dennis Andriesse, Herbert Bos, Cristiano Giufrida, and Gernot Heiser. Benchmarking crimes:an emerging threat in systems security. `https://ts.data61.csiro.au/publications/papers/Kouwe_ABGH_18:arxiv.pdf`, 2019.

[191] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clementine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1675–1689, New York, NY, USA, 2016. ACM.

[192] Marie Vasek and Tyler Moore. Do malware reports expedite cleanup? an experimental study. In *Presented as part of the 5th Workshop on Cyber Security Experimentation and Test*, Bellevue, WA, 2012. USENIX.

[193] Michael Venable, Mohamed R. Chouchane, Md Enamul Karim, and Arun Lakhotia. Analyzing memory accesses in obfuscated x86 executables. In Klaus Julisch and Christopher Kruegel, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 1–18, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[194] Timothy Vidas and Nicolas Christin. Evading android runtime analysis via sandbox detection. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 447–458, New York, NY, USA, 2014. ACM.

[195] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Detecting malicious domain names using deep learning approaches at scale. *Journal of Intelligent & Fuzzy Systems*, 34(3):1355–1367, 2018.

[196] VirusTotal. Launching virustotal monitor, a service to mitigate false positives. `https://blog.virustotal.com/2018/06/vtmonitor-to-mitigate-false-positives.html`, 2018.

[197] Thomas Vissers, Jan Spooren, Pieter Agten, Dirk Jumpertz, Peter Janssen, Marc Van Wesemael, Frank Piessens, Wouter Joosen, and Lieven Desmet. Exploring the ecosystem of malicious domain registrations in the. eu tld. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 472–493. Springer, 2017.

[198] Stijn Volckaert, Bart Coppens, and Bjorn De Sutter. Cloning your gadgets: Complete rop attack immunity with multi-variant execution. *IEEE Transactions on Dependable and Secure Computing*, 13(4):437–450, 2016.

[199] Ruoyu Wang. Ndss workshop on binary analysis research (bar) 2019. `https://ruoyuwang.me/bar2019/`, 2019.

[200] Andrew G West and Aziz Mohaisen. Metadata-driven threat classification of network endpoints appearing in malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 152–171. Springer, 2014.

[201] Andrew G. West and Aziz Mohaisen. Metadata-driven threat classification of network endpoints

appearing in malware. In Sven Dietrich, editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 152–171, Cham, 2014. Springer International Publishing.

[202] Carsten Willems, Felix C. Freiling, and Thorsten Holz. Using memory management to detect and extract illegitimate code for malware analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 179–188, New York, NY, USA, 2012. ACM.

[203] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, 5(2):32–39, 2007.

[204] Zhenyu Wu, Steven Gianvecchio, Mengjun Xie, and Haining Wang. Mimimorphism: A new approach to binary code obfuscation. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 536–546. ACM, 2010.

[205] Mengjun Xie, Zhenyu Wu, and Haining Wang. Honeyim: Fast detection and suppression of instant messaging malware in enterprise-like networks. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 64–73. IEEE, 2007.

[206] Guanhua Yan, Nathan Brown, and Deguang Kong. Exploring discriminatory features for automated malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 41–61. Springer, 2013.

[207] Wei Yang, Deguang Kong, Tao Xie, and Carl A Gunter. Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 288–302. ACM, 2017.

[208] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 116–127. ACM, 2007.

[209] Akira Yokoyama, Kou Ishii, Rui Tanabe, Yinmin Papa, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, Daisuke Inoue, Michael Brengel, Michael Backes, et al. Sandprint: fingerprinting malware sandboxes to provide intelligence for sandbox evasion. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 165–187. Springer, 2016.

[210] Fei Zhang, Patrick PK Chan, Battista Biggio, Daniel S Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. *IEEE transactions on cybernetics*, 46(3):766–777, 2016.

[211] Hang Zhang, Dongdong She, and Zhiyun Qian. Android root and its providers: A double-edged sword. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1093–1104. ACM, 2015.

[212] Mu Zhang, Yue Duan, Heng Yin, and Zhiruo Zhao. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1105–1116. ACM, 2014.

[213] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*, pages 95–109. IEEE, 2012.

[214] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Chengyu Song, and Wei Zou. Collecting autonomous spreading malware using high-interaction honeypots. In *International Conference on Information and Communications Security*, pages 438–451. Springer, 2007.