2012 IEEE Symposium on Security and Privacy

Prudent Practices for Designing Malware Experiments: Status Quo and Outlook

Christian Rossow*‡, Christian J. Dietrich*, Chris Grier†§, Christian Kreibich†§,
Vern Paxson†§, Norbert Pohlmann*, Herbert Bos‡, Maarten van Steen‡
* *Institute for Internet Security, Gelsenkirchen*
{rossow,dietrich,pohlmann}@internet-sicherheit.de
§ *University of California, Berkeley*
† *International Computer Science Institute, Berkeley*
grier@icsi.berkeley.edu,{christian,vern}@icir.org
‡ *VU University Amsterdam, The Network Institute*
{herbertb,steen}@cs.vu.nl

***Abstract*—Malware researchers rely on the observation of malicious code in execution to collect datasets for a wide array of experiments, including generation of detection models, study of longitudinal behavior, and validation of prior research. For such research to reflect prudent science, the work needs to address a number of concerns relating to the correct and representative use of the datasets, presentation of methodology in a fashion sufficiently transparent to enable reproducibility, and due consideration of the need not to harm others.**

In this paper we study the methodological rigor and prudence in 36 academic publications from 2006–2011 that rely on malware execution. 40% of these papers appeared in the 6 highest-ranked academic security conferences. We find frequent shortcomings, including problematic assumptions regarding the use of execution-driven datasets (25% of the papers), absence of description of security precautions taken during experiments (71% of the articles), and oftentimes insufficient description of the experimental setup. Deficiencies occur in top-tier venues and elsewhere alike, highlighting a need for the community to improve its handling of malware datasets. In the hope of aiding authors, reviewers, and readers, we frame guidelines regarding transparency, realism, correctness, and safety for collecting and using malware datasets.

I. Introduction

Observing the host- or network-level behavior of malware as it executes constitutes an essential technique for researchers seeking to understand malicious code. Dynamic malware analysis systems like Anubis [8], CWSandbox [50] and others [16, 22, 27, 36, 42] have proven invaluable in generating ground truth characterizations of malware behavior. The anti-malware community regularly applies these ground truths in scientific experiments, for example to evaluate malware detection technologies [2, 10, 17, 19, 24, 26, 30, 33, 44, 48, 52–54], to disseminate the results of large-scale malware experiments [6, 11, 42], to identify new groups of malware [2, 5, 38, 41], or as training datasets for machine learning approaches [20, 34, 35, 38, 40, 41, 47, 55]. However, while analysis of malware execution clearly holds importance for the community, the data collection and subsequent analysis processes face numerous potential pitfalls.

In this paper we explore issues relating to prudent experimental evaluation for projects that use malware-execution datasets. Our interest in the topic arose while analyzing malware and researching detection approaches ourselves, during which we discovered that well-working lab experiments could perform much worse in real-world evaluations. Investigating these difficulties led us to identify and explore the pitfalls that caused them. For example, we observed that even a slight artifact in a malware dataset can inadvertently lead to unforeseen performance degradation in practice.

Thus, we highlight that performing prudent experiments involving such malware analysis is harder than it seems. Related to this, we have found that the research community's efforts (including ours) frequently fall short of fully addressing existing pitfalls. Some of the shortcomings have to do with presentation of scientific work, i.e., authors remaining silent about information that they could likely add with ease. Other problems, however, go more deeply, and bring into question the basic representativeness of experimental results.

As in any science, it is desirable for our community to ensure we undertake prudent experimental evaluations. We define experiments reported in our paper as *prudent* if they are are correct, realistic, transparent, and do not harm others. Such prudence provides a foundation for the reader to objectively judge an experiment's results, and only well-framed experiments enable comparison with related work. As we will see, however, experiments in our community's publications could oftentimes be improved in terms of transparency, e.g., by adding and explaining simple but important aspects of the experiment setup. These additions render the papers more understandable, and enable others to reproduce results. Otherwise, the community finds itself at risk of failing to enable sound confirmation of previous results.

In addition, we find that published work frequently lacks sufficient consideration of experimental design and empirical assessment to enable translation from proposed methodologies to viable, practical solutions. In the worst case, papers can validate techniques with experimental results that suggest the authors have solved a given problem, but

DOI 10.1109/SP.2012.14

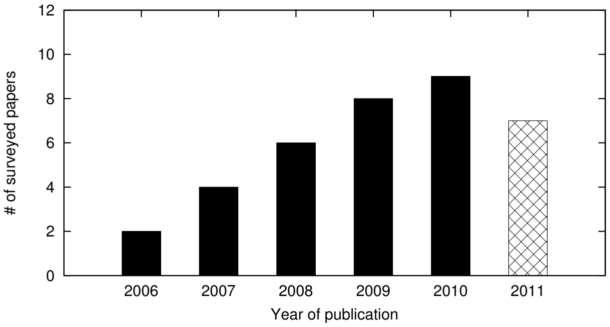

Figure 1: Surveyed papers using malware execution, per year.

the solution will prove inadequate in real use. In contrast, well-designed experiments significantly raise the quality of science. Consequently, we argue that it is important to have guidelines regarding both experimental design and presentation of research results.

We aim in this work to frame a set of guidelines for describing and designing experiments that incorporate such prudence, hoping to provide touchstones not only for authors, but also for reviewers and readers of papers based on analysis of malware execution. To do so, we define goals that we regard as vital for prudent malware experimentation: *transparency*, *realism*, *correctness*, and *safety*. We then translate these goals to guidelines that researchers in our field can use.

We apply these guidelines to 36 recent papers that make use of malware execution data, 40% from top-tier venues such as ACM CCS, IEEE S&P, NDSS and USENIX Security, to demonstrate the importance of considering the criteria. Figure 1 shows the number of papers we reviewed by publishing year, indicating that usage of such datasets has steadily increased. Table II (on page 6) lists the full set of papers. We find that almost all of the surveyed papers would have significantly benefited from considering the guidelines we frame, indicating, we argue, a clear need for more emphasis on rigor in methodology and presentation in the subfield. We also back up our assessment of the significance of some of these concerns by a set of conceptually simple experiments performed using publicly available datasets.

We acknowledge that fully following the proposed guidelines can be difficult in certain cases, and indeed this paper comes up short in some of these regards itself. For example, we do not fully transparently detail our survey datasets, as we thought that doing so might prove more of a distraction from our overall themes than a benefit. Still, the proposed guidelines can—when applicable—help with working towards scientifically rigorous experiments when using malware datasets.

To summarize our contributions:

- We identify potential pitfalls when designing experiments based on malware execution, and estimate the impact of these pitfalls in a few experiments.
- We devise guidelines to help with designing and presenting scientifically rigorous experiments.
- Our survey of 36 papers shows that our community could better address a number of shortcomings in typical malware datasets by adhering to these guidelines.
- We show that, contrary to our expectations, most of the problems occur equally in publications in top-tier research conferences and in less prominent venues.

II. Designing Prudent Experiments

We begin by discussing characteristics important for prudent experimentation with malware datasets. In formulating these criteria, we draw inspiration from extensive experience with malware analysis and malware detection, as well as from lessons we have learned when trying to assess papers in the field and—in some cases—reproducing their results. We emphasize that our goal is not to criticize malware execution studies in general. Instead, we highlight pitfalls when using malware datasets, and suggest guidelines how to devise prudent experiments with such datasets.

We group the pitfalls that arise when relying on data gathered from malware execution into four categories. Needless to say, compiling *correct datasets* forms a crucial part of any experiment. We further experienced how difficult it proves to ensure *realism* in malware execution experiments. In addition, we must provide *transparency* when detailing the experiments to render them both repeatable and comprehensible. Moreover, we believe that legal and ethical considerations mandate discussion of how conduct such experiments *safely*, mitigating harm to others. For each of these four "cornerstones of prudent experimentation", we now outline more specific aspects and describe guidelines to ensure prudence. As we will show later, the following guidelines can be used by our community to overcome common shortcomings in existing experiments.

A. Correct Datasets

1) **Check if goodware samples should be removed from datasets.** Whereas *goodware* (legitimate software) has to be present for example in experiments to measure false alarms, it is typically not desirable to have goodware samples in datasets to estimate false negative rates. However, malware execution systems open to public sample submission lack control over whether specimens submitted to the system in fact consist of malware; the behavior of such samples remains initially *unknown* rather than *malicious* per se. (We explore this concern as one of our illustrative experiments in § V-B.) We advocate that researchers use sources of malware specimens gathered via means that avoid the possible presence of goodware; explicitly remove goodware samples from their datasets;

or compile sample subsets based on malware family labels.

2) **Balance datasets over malware families.** In unbalanced datasets, aggressively polymorphic malware families will often unduly dominate datasets filtered by sample-uniqueness (e.g., MD5 hashes). Authors should discuss if such imbalances biased their experiments, and, if so, balance the datasets to the degree possible.
3) **Check whether training and evaluation datasets should have distinct families.** When splitting datasets based on sample-uniqueness, two distinct malware samples of one family can potentially appear in both the training and validation dataset. Appearing in both may prove desirable for experiments that derive generic detection models for malware families by training on sample subsets. In contrast, authors designing experiments to evaluate on previously unseen malware types should *separate* the sets based on families.
4) **Perform analysis with higher privileges than the malware's.** Malware with rootkit functionality can interfere with the OS data structures that kernel-based sensors modify. Such malware can readily influence monitoring components, thus authors ought to report on the extent to which malware samples and monitoring mechanisms collide. For example, kernel-based sensors could monitor whenever a malware gains equal privileges by observing if it is loading a kernel driver. Ideally, sensors are placed at a level where they cannot be modified, such as monitoring system calls with a system emulator or in a VMM.
5) **Discuss and if necessary mitigate analysis artifacts and biases.** Execution environment artifacts, such as the presence of specific strings (e.g., user names or OS serial keys) or the software configuration of an analysis environment, can manifest in the specifics of the behavior recorded for a given execution. Particularly when deriving models to detect malware, papers should explain the particular facets of the execution traces that a given model leverages. Similarly, biases arise if the malware behavior in an analysis environment differs from that manifest in an infected real system, for example due to containment policies.
6) **Use caution when blending malware activity traces into benign background activity.** The behavior exhibited by malware samples executing in dynamic analysis environments differs in a number of ways from that which would manifest in victim machines in the wild. Consequently, environment-specific performance aspects may poorly match those of the background activity with which experimenters combine them. The resulting idiosyncrasies may lead to seemingly excellent evaluation results, even though the system will perform worse in real-world settings. Authors should consider these issues, and discuss them explicitly *if* they decide to blend malicious traces with benign background activity.

B. Transparency

1) **State family names of employed malware samples.** Consistent malware naming remain a thorny issue, but labeling the employed malware families in some form helps the reader identify for which malware a methodology works. As we illustrate in § V-C, employing a large number of unique malware *samples* does not imply family diversity, due to the potential presence of binary-level polymorphism. If page-size limitations do not allow for such verbose information, authors can outsource this information to websites and add references to their paper accordingly.
2) **List which malware was analyzed when.** To understand and repeat experiments the reader requires a summary, perhaps provided externally to the paper, that fully describes the malware samples in the datasets. Given the ephemeral nature of some malware, it helps to capture the dates on which a given sample executed to put the observed behavior in context, say of a botnet's lifespan that went through a number of versions or ended via a take-down effort.
3) **Explain the malware sample selection.** Researchers oftentimes study only a subset of all malware specimens at their disposal. For instance, for statistically valid experiments, evaluating only a *random selection* of malware samples may prove necessary. Focusing on more recent analysis results and ignoring year-old data may increase relevance. In either case, authors should describe how they selected the malware subsets, and if not obvious, discuss any potential bias this induces. Note that random sample selections still may have imbalances that potentially need to be further addressed (see guideline A.2).
4) **Mention the system used during execution.** Malware may execute differently (if at all) across various systems, software configurations and versions. Explicit description of the particular system(s) used (e.g., "Windows XP SP3 32bit without additional software installations") renders experiments more transparent, especially as presumptions about the "standard" OS change with time. When relevant, authors should also include version information of installed software.
5) **Describe the network connectivity of the analysis environment.** Malware families assign different roles of activity depending on a system's connectivity, which can significantly influence the recorded behavior. For example, in the Waledac botnet [46], PCs connected via NAT primarily sent spam, while systems with public IP addresses acted as fast-flux "repeaters".

6) **Analyze the reasons for false positives and false negatives.** False classification rates alone provide little *clarification* regarding a system's performance. To reveal fully the limitations and potential of a given approach in other environments, we advocate thoughtful exploration of what led to the observed errors. Sommer and Paxson explored this particular issue in the context of anomaly detection systems [43].
7) **Analyze the nature/diversity of true positives.** Similarly, true positive rates alone often do not adequately reflect the potential of a methodology [43]. For example, a malware detector flagging hundreds of infected hosts may sound promising, but not if it detects only a single malware family or leverages an environmental artifact. Papers should evaluate the diversity manifest in correct detections to understand to what degree a system has general discriminative power.

C. Realism

1) **Evaluate relevant malware families.** Using significant numbers of popular malware families bolsters the impact of experiments. Given the ongoing evolution of malware, exclusively using older or sinkholed specimens can undermine relevance.
2) **Perform real-world evaluations.** We define a real-world experiment as an evaluation scenario that incorporates the behavior of a significant number of hosts in active use by people other than the authors. Real-world experiments play a vital role in evaluating the gap between a method and its application in practice.
3) **Exercise caution generalizing from a single OS version, such as Windows XP.** For example, by limiting analysis to a single OS version, experiments may fail with malware families that solely run or exhibit different behavior on disregarded OS versions. For studies that strive to develop results that generalize across OS versions, papers should consider to what degree we can generalize results based on one specific OS version.
4) **Choose appropriate malware stimuli.** Malware classes such as keyloggers require triggering by specific stimuli such as keypresses or user interaction in general. In addition, malware often expose additional behavior when allowed to execute for more than a short period [42]. Authors should therefore describe why the analysis duration they chose suffices for their experiments. Experiments focusing on the initialization behavior of malware presumably require shorter runtimes than experiments that aim to detect damage functionality such as DoS attacks.
5) **Consider allowing Internet access to malware.** Deferring legal and ethical considerations for a moment, we argue that experiments become significantly more realistic if the malware has Internet access. Malware often requires connectivity to communicate with command-and-control (C&C) servers and thus to expose its malicious behavior. In exceptional cases where experiments in simulated Internet environments are appropriate, authors need to describe the resulting limitations.

D. Safety

1) **Deploy and describe containment policies.** Well-designed containment policies facilitate realistic experiments while mitigating the potential harm malware causes to others over time. Experiments should at a minimum employ basic containment policies such as redirecting spam and infection attempts, and identifying and suppressing DoS attacks. Authors should discuss the containment policies and their implications on the fidelity of the experiments. Ideally, authors also monitor and discuss security breaches in their containment.

III. Methodology for Assessing the Guidelines

The previous section described guidelines for designing and presenting scientifically prudent malware-driven experiments. As an approach to verify if our guidelines are in fact useful, we analyzed in which cases they would have significantly improved experiments in existing literature. This section describes our methodology for surveying relevant publications with criteria derived from our guidelines.

A. Assessment Criteria

Initially, we establish a set of criteria for assessing the degree to which experiments presented in our community adhere to our guidelines. We aim to frame these assessments with considerations of the constraints the reviewer of a paper generally faces, because we ultimately wish to gauge how well the subfield develops its research output. Consequently, we decided not to attempt to review source code or specific datasets, and refrained from contacting individual authors to clarify details of the presented approaches. Instead, our goal is to assess the prudence of experiments given all the information available in a paper or its referenced related work, but no more. We employed these constraints since they in fact reflect the situation that a reviewer faces. A reviewer typically is not supposed to clarify missing details with the authors (and in the case of double-blind submissions, lacks the means to do so). That said, we advocate that readers facing different constraints should contact authors to clarify lacking details whenever possible.

Table I lists the guideline criteria we used to evaluate the papers. We translate each aspect addressed in § II into at least one concrete check that we can perform when

Criterion	Gdl.	Imp.	Description
Correct Datasets			
Removed goodware	A.1)	●	Removed legitimate binaries from datasets
Avoided overlays	A.6)	●	Avoided comparison of execution output with real system output
Balanced families	A.2)	◐	Training datasets balanced in terms of malware families, not individual specimens
Separated datasets	A.3)	◐	Appropriately separated training and evaluation datasets based on families
Mitigated artifacts/biases	A.5)	◐	Discussed and if necessary mitigated analysis artifacts or biases
Higher privileges	A.4)	◐	Performed analysis with higher privileges than the malware
Transparency			
Interpreted FPs	B.6)	●	Analyzed when and why the evaluation produced false positives
Interpreted FNs	B.6)	●	Analyzed when and why the evaluation produced false negatives
Interpreted TPs	B.7)	●	Analyzed the nature/diversity of true positives
Listed malware families	B.2)	◐	Listed the family names of the malware samples
Identified environment	B.4)	◐	Named or described the execution environment
Mentioned OS	B.4)	◐	Mentioned the operating system used during execution analysis
Described naming	B.1)	◐	Described the methodology of how malware family names were determined
Described sampling	B.3)	○	Mentioned the malware sample selection mechanism
Listed malware	B.1)	○	Listed which malware was when analyzed
Described NAT	B.5)	○	Described whether NAT was used or not
Mentioned trace duration	C.4)	○	Described for how long malware traces were recorded.
Realism			
Removed moot samples	C.1)	●	Explicitly removed outdated or sinkholed samples from dataset
Real-world FP exp.	C.2)	●	Performed real-world evaluation measuring wrong alarms/classifications
Real-world TP exp.	C.2)	●	Performed real-world evaluation measuring true positives
Used many families	C.1)	●	Evaluated against a significant number of malware families
Allowed Internet	C.5)	◐	Allowed Internet access to malware samples
Added user interaction	C.4)	○	Explicitly employed user interaction to trigger malware behavior
Used multiple OSes	C.3)	○	Analyzed malware on multiple operating systems
Safety			
Deployed containment	D.1)	●	Deployed containment policies to mitigate attacks during malware execution

Table I: List of guideline criteria assessed during our survey. The second column denotes the guidelines from which we derived this criterion. The third column denotes the importance that we devote to this subject: ● is a must, ◐ should be done, and ○ is nice to have.

reading a given paper.[1] We defined the assessment criteria in an objective manner such that each item can be answered without ambiguity. We also assign a three-level qualitative importance rating to each check, based on our experience with malware execution analysis. Later on, this rating allows us to weigh the interpretation of the survey results according to the criteria's criticality levels.

For an informal assessment of our approach, we asked the authors of two papers to apply our criteria.[2] The researchers were asked if the criteria were applicable, and if so, if the criteria were met in their own work. During this calibration process, we broadened the check to determine coverage of false positives and false negatives, to allow us to perform a generic assessment. In addition, as we will discuss later, we realized that not all criteria can be applied to all papers.

B. *Surveyed Publications*

We assessed each of the guideline criteria against the 36 scientific contributions ("papers") in Table II. We obtained this list of papers by systematically going through all of the proceedings of the top-6 computer- and network-security conferences from 2006–2011. [3] We added a paper to our list if any of its experiments make use of PC malware execution-driven datasets. We then also added an arbitrary selection of relevant papers from other, less-prestigious venues, such that in total about two fifth (39%) of the 36 surveyed papers were taken from the top-6 security conferences. As Figure 1 shows, we see increasing use of malware execution during recent years.

The surveyed papers use malware datasets for diverse purposes. A significant number used dynamic analysis results as input for a *training process* of malware detection methods. For example, Botzilla [40] and Wurzinger

[1] Although the guideline "Choose appropriate malware stimuli" is in the *Realism* section, we added the criterion "Mentioned trace duration" (as one possible criterion for this guideline) to the *Transparency* category.

[2] One of these is a coauthor of this paper, too. However, he undertook applying the criteria prior to obtaining any knowledge of the separate assessment of his paper made as part of our survey.

[3] We determined the top-6 conferences based on three conference-ranking websites: (1) Microsoft Academic Search - Top Conferences in Security & Privacy (http://academic.research.microsoft.com/RankList?entitytype=3&topdomainid=2&subdomainid=2), (2) Guofei Gu's Computer Security Conference Ranking and Statistic (http://faculty.cs.tamu.edu/guofei/sec_conf_stat.htm), and (3) Jianying Zhou's Top Crypto and Security Conferences Ranking (http://icsd.i2r.a-star.edu.sg/staff/jianying/conference-ranking.html). As all rankings agreed on the top 6, we chose those as constituting top-tier conferences: ACM CCS, IEEE S&P, NDSS, USENIX Security, and two conferences (Crypto and Eurocrypt) without publications in our focus. We defined this list of top-venues prior to assembling the list of papers in our survey.

#	Authors	Venue	Top	Title
1	Lanzi et al. [30]	ACM CCS 2010	✓	AccessMiner: Using System-Centric Models for Malware Protection
2	Morales et al. [35]	IEEE SecureComm 2010		Analyzing and Exploiting Network Behaviors of Malware
3	Rieck et al. [41]	Journal of Comp. Sec.		Automatic Analysis of Malware Behavior using Machine Learning
4	Bailey et al. [2]	RAID 2007		Automated Classification and Analysis of Internet Malware
5	Wurzinger et al. [51]	ESORICS 2009		Automatically Generating Models for Botnet Detection
6	Bayer et al. [6]	USENIX LEET 2009		A View on Current Malware Behaviors
7	Perdisci et al. [38]	NSDI 2010		Behavioral Clustering of HTTP-Based Malware and Signature Generation [...]
8	Kirda et al. [24]	USENIX Security 2006	✓	Behavior-based spyware detection
9	Jang et al. [21]	ACM CCS 2011	✓	BitShred: Feature Hashing Malware for Scalable Triage and Semantic Analysis
10	Zhang et al. [54]	ASIACCS 2011		Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling
11	Gu et al. [17]	USENIX Security 2008	✓	BotMiner: Clustering Analysis of Network Traffic for [...] Botnet Detection
12	Strayer et al. [48]	Adv. in Info. Sec. 2008		Botnet Detection Based on Network Behavior
13	Gu et al. [19]	NDSS 2008	✓	BotSniffer: Detecting Botnet C&C Channels in Network Traffic
14	Bowen et al. [10]	RAID 2010		BotSwindler: Tamper Resistant Injection of Believable Decoys [...]
15	Liu et al. [33]	ISC 2008		BotTracer : Execution-based Bot-like Malware Detection
16	Rieck et al. [40]	ACM SAC 2010		Botzilla: Detecting the "Phoning Home" of Malicious Software
17	Stinson et al. [44]	DIMVA 2007		Characterizing Bots' Remote Control Behavior
18	Lindorfer et al. [32]	RAID 2011		Detecting Environment-Sensitive Malware
19	Gu et al. [18]	USENIX Security 2007	✓	Detecting Malware Infection Through IDS-Driven Dialog Correlation
20	Caballero et al. [12]	ACM CCS 2009	✓	Dispatcher: Enabling Active Botnet Infiltration [...]
21	Kolbitsch et al. [25]	USENIX Security 2009	✓	Effective and Efficient Malware Detection at the End Host
22	Balzarotti et al. [3]	NDSS 2010	✓	Efficient Detection of Split Personalities in Malware
23	Stone-Gross et al. [47]	ACSAC 2009		FIRE: FInding Rogue nEtworks
24	Bayer et al. [7]	ACM SAC 2010		Improving the Efficiency of Dynamic Malware Analysis
25	Kolbitsch et al. [26]	IEEE S&P 2010	✓	Inspector Gadget: Automated Extraction of Proprietary Gadgets [...]
26	Jacob et al. [20]	USENIX Security 2011	✓	JACKSTRAWS: Picking Command and Control Connections from Bot Traffic
27	Rieck et al. [39]	DIMVA 2008		Learning and Classification of Malware Behavior
28	Caballero et al. [11]	USENIX Security 2011	✓	Measuring Pay-per-Install: The Commoditization of Malware Distribution
29	Yu et al. [53]	Journal of Networks 2010		Online Botnet Detection Based on Incremental Discrete Fourier Transform
30	Comparetti et al. [15]	IEEE S&P 2009	✓	Prospex: Protocol Specification Extraction
31	Rossow et al. [42]	BADGERS 2011		Sandnet: Network Traffic Analysis of Malicious Software
32	Bayer et al. [5]	NDSS 2009	✓	Scalable, Behavior-Based Malware Clustering
33	Barford et al. [4]	USENIX HotBots 2007		Toward Botnet Mesocosms
34	Yen et al. [52]	DIMVA 2008		Traffic Aggregation for Malware Detection
35	Zhu et al. [55]	SecureComm 2009		Using Failure Information Analysis to Detect Enterprise Zombies
36	Livadas et al. [34]	IEEE LCN 2006		Using Machine Learning Techniques to Identify Botnet Traffic

Table II: List of surveyed papers ordered by title. We shorten some titles with "[...]" due to space limitations.

et al. [51] use malicious network traffic to automatically generate payload signatures of malware. Similarly, Perdisci et al. [38] propose a methodology to derive signatures from malicious HTTP request patterns. Livadas et al. [34] identify IRC-based C&C channels by applying machine-learning techniques to malware execution results. Zhu et al. [55] train SVMs to model the abnormally high network failure rates of malware. Morales et al. [35] manually derive characteristics from malware observed during execution to create detection signatures. Malheur [39, 41] can cluster and classify malware based on ordered behavioral instructions as observed in CWSandbox. Kolbitsch et al. [25] present a host-based malware detection mechanism relying on system call slices as observed in Anubis.

In addition, we have surveyed papers that used malware execution solely to *evaluate methodologies*. Most of these papers leverage malware traces to measure true positive rates of malware detection mechanisms [10, 17–19, 24, 30, 33, 44, 48, 52–54]. Typically, the authors executed malware samples in a contained environment and used the recorded behavior as ground truth for malicious behavior, either via network traces (for assessing network-based IDSs) or via host behavior such as system call traces (for system-level approaches). Similarly, researchers have used malware execution traces for evaluating methodologies to understand protocol semantics [12, 15], to extract isolated code parts from malware binaries [26], to detect if malware evades contained environments [3], or to improve the efficiency of dynamic analysis [7].

A third group of papers used malware traces to *obtain a better understanding of malware behavior*. For example, JACKSTRAWS [20] leverages Anubis to identify botnet C&C channels. Similarly, FIRE [47] identifies rogue networks by analyzing malware communication endpoints. Caballero et al. [11] execute malware to measure the commoditization of pay-per-install networks. DISARM [32] measures how different malware behaves in virtualized environments compared to Anubis. Bayer et al. [5] and Jang et al. [21] present efficient clustering techniques for malware behavior. Bailey et al. [2] label malware based on its behavior over time. Finally, Bayer et al. [6] and Rossow et al. [42] analyze the behavioral profiles of malware samples as observed in Anubis and Sandnet.

C. *Survey Methodology*

To ensure consistency and accuracy in our survey results, two of our authors conducted an initial survey of the full set of papers. Employing a fixed pair of reviewers helps to ensure that all papers received the same interpretation of the guideline criteria. When the two reviewers did not agree, a third author decided on the specific case. In general, if in doubt or when encountering vague decisions, we classified the paper as conforming with the guideline ("benefit of the doubt"). Note that our assessments of the papers contain considerably more detail than the simple statistic summaries presented here. If a paper lacked detail regarding experimental methodology, we further reviewed other papers or technical reports describing the particular malware execution environment. We mark criteria results as "unknown" if after doing so the experimental setup remained unclear.

We carefully defined subsets of *applicable papers* for all criteria. For instance, executions of malware recompiled to control network access do not require containment policies. Similarly, analyzing the diversity of false positives only applies to methodologies that have false positives, while removing goodware samples only matters when relying on unfiltered datasets with unknown (rather than guaranteed malicious) binaries. Also, removing outdated or sinkholed samples might not apply if the authors manually assembled their datasets. Balancing malware families is applicable only for papers that use datasets in classification experiments and if authors average classification performances over the (imbalanced) malware samples. Moreover, we see a need to separate datasets in terms of families only if authors suggest that a methodology performs well on previously unseen malware types. We further define real-world experiments to be applicable only for malware detection methodologies. These examples show that building subsets of applicable papers is vital to avoid skew in our survey results. Consequently, we note for all criteria the number of papers to which we deemed they applied.

We also sometimes found it necessary to interpret criteria selectively to papers. For example, whereas true-positive analysis is well-defined for assessing malware detection approaches, we needed to consider how to translate the term to other methodologies (e.g., malware clustering or protocol extraction). Doing so enabled us to survey as many applicable papers as possible, while keeping the criteria fairly generic and manageable. In the case of malware clustering techniques, we translated *recall* and *precision* to true positive and false positive rate, respectively. This highlights the difficulty of arriving at an agreed-upon set of guidelines for designing prudent experiments.

IV. Survey Observations

We divide our survey interpretation into three parts. First, in a *per-guideline analysis*, we discuss to which extent specific guidelines were met. The subsequent *per-paper analysis* assesses whether only a small fraction of all papers accounts for the results, or if our findings hold more generally across all of the papers. Finally, a *top-venue analysis* details how papers appearing in more competitive research venues (as previously defined) compare with those appearing in other venues.

A. *Per-Guideline Analysis*

Table III lists the results of our assessment methodology ordered by theme and importance. The second major column includes statistics on all surveyed papers, while the third major column represents data from publication at top-tier venues only. *App* specifies the number of papers for which the criterion applied. *OK* states the proportion of those applicable papers that adhered to the guideline, whereas *Ukwn* specifies the proportion for which we could not assess the guideline due to lack of experimental description.

1) Correctness: In this section we highlight instances of criteria that potentially call into question the basic correctness of a paper's results.

In five cases, we find papers that mix behavioral traces taken from malware execution with traces from real systems. We find it difficult to gauge the degree of realism in such practices, since malware behavior recorded in an execution environment may deviate from the behavior exhibited on systems infected in the wild. For instance, Celik et al. [13] have pointed out that time-sensitive features such as frames per hour exhibit great sensitivity to the local network's bandwidth and connectivity latency; blending malware flows into other traces thus requires great care in order to avoid unnatural heterogeneity in those features. Another difference is generally the lack of user interaction in malware execution traces, which typically exists in real system traces. Consequently, we argue that researchers should not base real-world evaluations on mixed (*overlay*) datasets. On the positive side, two papers avoided overlay datasets and instead deployed sensors to large networks for real-world evaluations [38, 40].

In two papers, the authors present new findings on malware behavior derived from datasets of public dynamic analysis environments, but did not remove goodware from such datasets. Another two malware detection papers include potentially biased false negative experiments, as the datasets used for these false negative evaluations presumably contain goodware samples. We illustrate in § V-B that a significant ratio of samples submitted to public execution environments consists of goodware. Other than these four papers, all others filtered malware samples using anti-virus labels. However, no author discussed removing outdated or sinkholed malware families from the datasets, which has significantly side-effects in at least one such case.

Summarizing, at least nine (25%) distinct papers appear to suffer from *clearly significant* problems relating to our three most basic *correctness* criteria. In addition, observing the range of further potential pitfalls and the survey results, we

Criterion	Imp.	All Papers			Top-Venue Papers			Description
Correctness		App	Ukwn	OK	App	Ukwn	OK	
Removed goodware	●	9	0%	44%	4	0%	50%	More than half potentially include experiments with goodware samples in the datasets. In these cases, authors seem to have mistakenly presumed binaries from public binary execution environments as malicious.
Avoided overlays	●	7	0%	29%	4	0%	0%	Five of the seven papers that perform real-world experiments to measure true positives merged traces from execution environments into real-world ones.
Balanced families	◐	13	0%	54%	2	0%	50%	Only half of the papers considered balancing training datasets based on malware families rather than individual specimens, possibly biasing the detection models or testing datasets towards polymorphic families.
Separated datasets	◐	8	0%	0%	1	0%	0%	No paper discussed issues regarding separating training and testing datasets in terms of malware families. This may invalidate experiments testing if a methodology is generic.
Mitigated artifacts/biases	◐	36	0%	28%	14	0%	50%	Less than a third discussed or removed artifacts/biases from the datasets. If present, such artifacts/biases could significantly influence experimental validity; only real-world assessment can prove otherwise.
Higher Privileges	◐	36	6%	75%	14	0%	86%	The quarter of papers that use data recorded at a privilege level equal to that of the malware execution risk increased evasion.
Transparency		App	Ukwn	OK	App	Ukwn	OK	
Interpreted FPs	●	25	n/a	64%	9	n/a	89%	Of the papers that present false positive rates, a third lacks details beyond the plain numbers.
Interpreted FNs	●	21	n/a	48%	7	n/a	57%	In more than half of the cases, readers have to speculate why false negatives occur.
Interpreted TPs	●	30	n/a	60%	11	n/a	55%	Two out of five applicable papers do not interpret true positives. This omission can hide vital information on the *basis* and *diversity* of classifications.
Listed malware families	◐	36	n/a	81%	14	n/a	86%	Most papers adequately name the malware families in their datasets. Seven papers rely on high numbers of distinct samples instead, hiding on which families experiments are based.
Identified environment	◐	36	n/a	81%	14	n/a	79%	A minority of papers fail to name or describe the execution environment used to capture malware traces used during experiments.
Mentioned OS	◐	36	n/a	64%	14	n/a	64%	A third do not mention the OS used during their experiments.
Described naming	◐	32	n/a	50%	12	n/a	58%	Only half described how the family labels for malware samples were obtained.
Describe sampling	○	16	n/a	81%	5	n/a	60%	A fifth of the papers using bulks of malware samples do not motivate how the subsets from all available reports of a dynamic analysis environment were chosen.
Listed malware	○	36	n/a	11%	14	n/a	7%	Almost all papers lack details on which particular malware samples (e.g., distinct MD5 hashes) were analyzed.
Described NAT	○	30	n/a	10%	11	n/a	9%	Only three papers mention whether the execution environment used NAT or if the infected machine was assigned a public IP addresses.
Mentioned trace duration	○	36	n/a	64%	14	n/a	57%	A third do not mention for how long malware executed when capturing traces.
Realism		App	Ukwn	OK	App	Ukwn	OK	
Removed moot samples	●	16	0%	0%	5	0%	0%	No paper discussed excluding execution of outdated malware binaries or those with sinkholed communications. As we illustrate in § V-D, such traces can make up a significant fraction of recorded malware behavior.
Real-world FP exp.	●	20	0%	50%	6	0%	67%	Only half of the malware detection papers include real-world false positive experiments, vital for prudently evaluating the overhead of wrong alarms.
Real-world TP exp.	●	20	0%	35%	6	0%	67%	Most of the malware detection papers lack real-world true positive experiments.
Used many families	●	36	1/8/745		14	1/8/745		Minimum/median/maximum number of malware families used in experiments.
Allowed Internet	◐	36	6%	75%	14	0%	79%	A fifth of the papers either simulated the Internet or modified bot source code to run without it, raising concerns of the realism of experiments.
Added user interaction	○	36	0%	3%	14	0%	0%	In only one case the authors explicitly deployed sophisticated user interactions to trigger certain malware behavior. The lack of such mention in other papers may indicate that experiments lack user-triggered behaviors such as keylogging.
Used multiple OSes	○	36	22%	19%	14	21%	29%	Only about a fifth seemed to deploy their experiments on multiple OSes.
Safety		App	Ukwn	OK	App	Ukwn	OK	
Deployed containment	●	28	71%	21%	11	64%	27%	The majority of papers did not explicitly mention containment policies, and 77% lack a policy description. This compromises transparency, and hinders readers to judge if authors gave sufficient consideration to mitigating malware attacks.

Table III: Overview and short interpretation of survey results.

speculate that more papers may suffer from other significant biases. For example, in another 15 cases, the authors did not explicitly discuss the presence/absence of sinkholed or inactive malware samples. In addition, three malware detection papers do not name malware families, but instead use a diverse set of malware binaries during experiments. We illustrate in § V-C that such datasets are typically biased and potentially miss significant numbers of malware families. We further observed seven papers with experiments based on machine learning that did not employ cross-validation and thus potentially failed to generalize the evaluation to other datasets. To name good examples, the authors in [25, 32, 39, 41] chose a subset of malware families and balanced the number of samples per family prior to the training process. Similarly, we observed authors performing cross-validation to avoid overfitting detection models [30, 39, 41, 51].

Lastly, nearly all of the papers omitted discussion of possible biases introduced by malware execution, such as malware behavior that significantly differs if binaries execute in a virtual machine [3, 32]. Typically, further artifacts or biases, for example, due to containment policies exist when executing malware as illustrated in § V-E. We highlight the importance of real-world scenarios, as they favor methodologies which evaluate against realistic and correct datasets.

2) Transparency: We observed two basic problems regarding transparent experiment descriptions in our community. First, descriptions of experimental setups lack sufficient detail to ensure repeatability. For example, 20% of the papers do not name or describe the execution environment. For a third of the papers it remains unclear on which OS the authors tested the proposed approach, and about a fifth do not name the malware families contained in the datasets. Consequently, in the majority of cases the reader cannot adequately understand the experimental setup, nor can fellow researchers hope to repeat the experiments. In addition, 75% do not describe containment policies.

Second, we find the majority of papers incompletely describe experimental results. That is, papers frequently fail to interpret the numeric results they present, though doing so is vital for effectively understanding the import of the findings. Consider the simple case of presenting detection rates. In which exact cases do false positives occur? Why do some malware families raise false negatives while others do not? Do the true positives cover sufficient behavioral diversity?

3) Realism: Our survey reveals that only a minority of papers includes real-world evaluations, and very few papers offer significant sample sizes (e.g., in numbers of hosts) for such experiments. The lack of real-world experiments makes it hard to judge whether a proposed methodology will also work in practice. We find that authors who *do* run real-world experiments often use locally accessible networks (e.g., a university campus, or a research lab). Doing so does not constitute a problem *per se*, but authors often base such experiments on the untenable assumption that these environments do not contain malware activity. In eight cases, authors used university networks for a false positive analysis only, although their methodology should also detect malware in such traces.

We noted a further eight papers that model malicious behavior on malware samples controlled by the authors themselves. Without justification, it seems unlikely that such malware samples behave similarly to the same samples when infecting victim machines in the wild. The malware execution environment may introduce further biases, e.g. via author-controlled servers that may exhibit unrealistically deterministic communication patterns. All of these cases lack representative real-world evaluations, which could have potentially offset these criteria.

We find that the typical paper evaluates its methodology against eight (median) distinct malware families, and five (14%) evaluated using only a single family. Similarly, two thirds of the surveyed malware detection methodologies evaluated against eight or fewer families. There may be a good reason for not taking into account further families, e.g., if no other malware families are applicable for a specific experiment. In general, however, we find it difficult to gauge whether such experiments provide statistically sound results that can generalize.

4) Safety: Most papers did not deploy or adequately describe containment. More than two thirds (71%) completely omit treatment of any containment potentially used during the experiments. The reasons for this may be that authors rely on referencing to technical reports for details on their containment solution. We found, however, that only few such reports detail the containment policies in place. Two papers state that the authors explicitly refrained from deploying containment policies.

B. Per-Paper Analysis

The preceding discussion has shown the high potential of our guidelines for improving specific prudence criteria. As a next step, we analyze how many papers can in total benefit from significant improvements.

To do so, Figure 2 details how many of the most important criteria (● in Table I)[4] a paper violated. The fewer criteria a paper met, the more its experiments could have been improved by using our guidelines. The figure shows that only a single paper fulfilled all of the applicable guidelines. More than half (58%) of the papers violate three or more criteria. In general, the plot shows a correlation between the number of violated criteria and the number of applicable criteria. This means that our guidelines become increasingly important when designing more complex experiments.

We then separate the results into presentation and safety issues (middle graph) and incorrect or unrealistic exper-

[4] We note that we devised the importance ranking prior to conducting the analyses in this section.

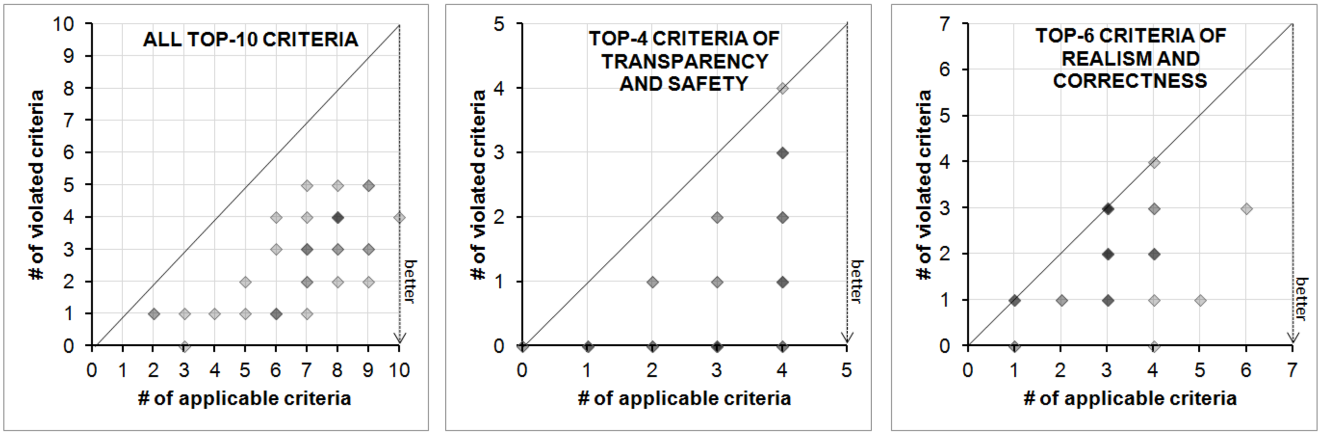

Figure 2: Guideline violations related to applicable criteria, separated into (1) all 10 most important criteria (left), (2) transparency/safety (middle), and (3) correctness/realism (right). Each dot represents one paper, darker dots cover more papers.

iments (right graph). We find that lacking transparency and safety constitutes a problem in half of the cases. Far more papers (92%) have deficiencies in establishing correct datasets and realistic experiments. Note that this does not imply that the experiments suffer from heavy flaws. It does flag, however, that many papers remain silent about important experimental descriptions. In addition, this analysis shows that experiments in applicable papers could be significantly improved in terms of correct datasets and realistic experiments.

In some cases, malware datasets were reused in related papers (such as [21]), often inheriting problems from the original experiments. In such cases, issues are mostly with the original paper. However, we knowingly did not remove such papers, as we wanted to survey the *use* instead of the *creation* of malware datasets.

C. Top-Venue Analysis

We now ask ourselves if experiments presented at top-tier conferences appear to be more prudent than others. To measure this, Figure 3 compares results for the ten most important guidelines (● in Table I). We do not observe any obvious prudence tendency towards top-tier conferences or other venues. The first strong difference regards the prevalence of real-world experiments: while more papers presented at top-tier venues include real-world scenarios, authors base these on potentially skewed overlay datasets (e.g., mixing malware traces in real traces). Second, we observed more papers interpreting false positives at top-tier conferences than at other venues. However, while the number and ratios of violations slightly differ across the criteria, the violations generally remain comparable. We therefore conclude that research published in top-tier conferences would equally benefit from our guidelines as papers presented at other venues. Thus, these shortcomings appear endemic to our field, rather than emerging as a property of less stringent peer review or the quality of submitted works.

V. Experiments

We now conduct four experiments that test four hypotheses we mentioned in previous sections. In particular, we will analyze the presence of (1) goodware, (2) malware family imbalances, (3) inactive and sinkholed samples, and (4) artifacts in malware datasets taken from contained environments that accept public submissions. Similar datasets were used in many surveyed experiments, raising the significance of understanding pitfalls with using such datasets. As we will show, our illustrative experiments underline the importance of proper experiment design and careful use of malware datasets. At the same time, these experiments show how we can partially mitigate some of the associated concerns.

A. Experimental Setup

We conducted all malware execution experiments in Sandnet [42], using a Windows XP SP3 32bit virtual machine connected to the Internet via NAT. We deploy containment policies that redirect harmful traffic (e.g., spam, infections) to local honeypots. We further limit the number of concurrent connections and the network bandwidth to mitigate DoS activities. An in-path honeywall NIDS watched for security breaches during our experiments. Other protocols (e.g., IRC, DNS or HTTP) were allowed to enable C&C communication. The biases affecting the following experiments due to containment should thus remain limited. We did not deploy user interaction during our experiments. As Windows XP malware was most prevalent among the surveyed papers, we did not deploy other OS versions during dynamic analysis.

We base experiments V-B and V-C on 44,958 MD5 distinct malware samples and a diverse set of more than 100 malware families. We (gratefully) received these samples as a snapshot of samples submitted to a large public dynamic analysis environment during Jan.1–30, 2011. The

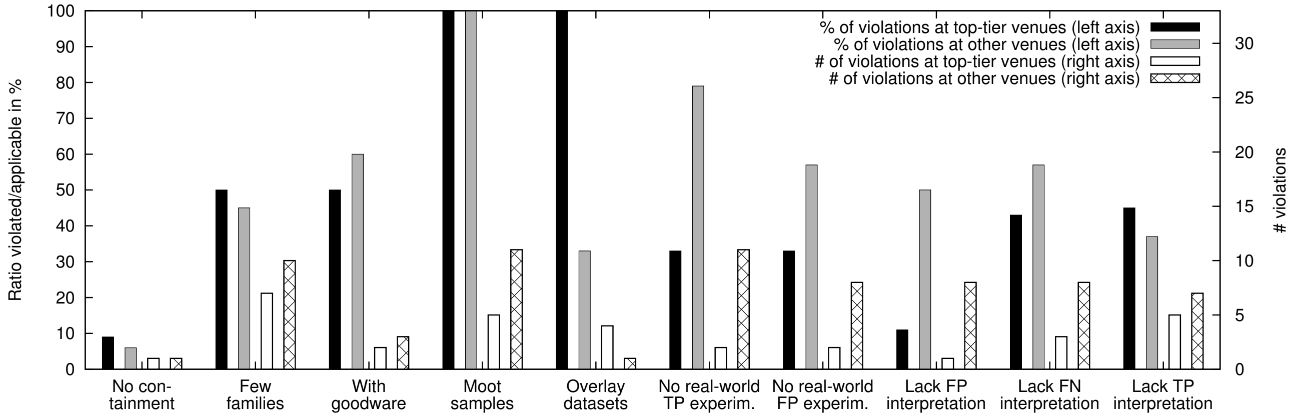

Figure 3: Violations at top-tier venues compared with other venues.

samples originated from a diverse set of contributors, including security companies, honeypot infrastructures, and spamtraps. To analyze the dynamic malware behavior in experiments V-E and V-D, we randomly chose 10,670 of these 44,958 samples. We executed this subset of samples and recorded the malware's network traces at the Internet gateway. An execution typically lasts for at least one hour, but for reasons of scale we stopped execution if malware did not show network activity in the first 15 minutes. The reader can find the data regarding execution date, trace duration, MD5 hashes, and family names of the malware samples used in the experiments at our website.[5] As we use the following experiments to measure the presence of imbalances, goodware, sinkholing and artifacts, we explicitly did not clean up our dataset in this regard.

B. Legitimate Samples Under Analysis

Experiments that erroneously consider legitimate software samples as malware suffer from bias. For example, when evaluating detection accuracies, legitimate software may cause false positives. Similarly, surveys of malicious behavior will exhibit bias if the underlying dataset contains legitimate software. Thus, in this experiment, we test our hypothesis that *goodware* is significantly present in public dynamic analysis systems' datasets.

To give lower bounds for the ratio of goodware, we queried the MD5 hash sum of all 44,958 binaries in two whitelists during the first week in November 2011. First, we query Shadowserver.org's *bin-test* [49] for known software. Second, we consulted *Bit9 Fileadvisor* [9], a file reputation mechanism also used by anti-spam vendors. *bin-test* revealed 176 (0.4%) goodware samples. The *Bit9 Fileadvisor* recognized 2,025 (4.5%) samples. In combination, both lists revealed 2,027 unique binaries as potentially being benign. As *Bit9* also includes malicious software in their database, we inspected a small sample of the 2,027 known binaries to estimate the ratio of goodware in the hits. In particular, we manually analyzed a subset of 100 randomly selected matches and found 78 to be legitimate software. Similarly, we cross-checked the 2,027 binaries via VirusTotal and found that 67.5% did not register any anti-virus detection. Estimating more conservatively, we use the minimum ratio of goodware samples (67.5%) to extrapolate the number of goodware samples within the 2,027 "whitelisted" samples. This translates to a *lower bound* of 1,366 (3.0%) goodware samples in our total dataset. We can also approximate an *upper bound* estimate regarding the prevalence of non-malicious samples by observing that 33% of the samples that were scanned by VirusTotal were not detected by any of the 44 vendors listed at VirusTotal. We therefore conclude that the ratio of legitimate binaries (3.0%–33%) may significantly bias experiments.

C. Distribution of Malware Families

In this experiment we test our hypothesis stating that polymorphic malware manifests in an unduly large proportion in randomly collected sets of malware samples. We used the VirusTotal labels obtained in Experiment V-B and counted the occurrences of malware families for each anti-virus vendor. To obtain the malware family names, we parsed the naming schemes of three anti-virus vendors (Avira, Kaspersky and Symantec) commonly used by our community to assign malware labels.

The CDF in Figure 4 shows the relationship of malware families to prevalences of families in our dataset. Ideally, a multi-family malware corpus stems from a uniform distribution, i.e., each malware family contributes the same number of samples. In our dataset, randomly collected from a large public dynamic analysis environment, we find this goal clearly violated: some malware families far dominate others. For example, when relying on Kaspersky, almost 80% of the malware samples belong to merely 10% of the

[5] See http://christian-rossow.de/publications/datasets/ieee12.htm

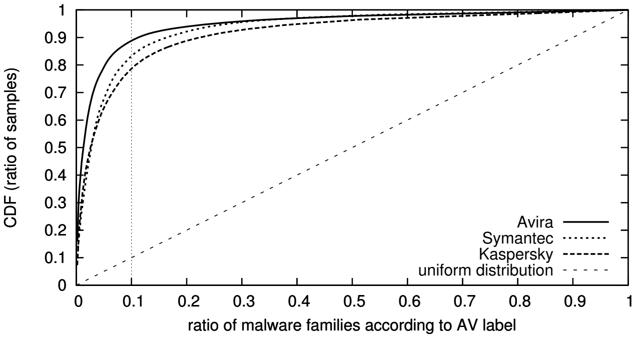

Figure 4: Family imbalances in randomly chosen malware samples.

families. In the worst case, this would mean that experiments performing well with 4/5's of the samples may not work with 90% of the remaining malware families. In summary, unless researchers take corresponding precautions, polymorphic malware families can disproportionately dominate randomly drawn corpora of malware samples.

D. Inactive or Sinkholed Samples

The identification of correctly functioning malware samples poses one of the major challenges of automated dynamic analysis. Large fractions of analyzed samples do not exhibit any behavior [42]. Further complicating things, even if network communication manifests, it remains unclear whether it constitutes successful operation and representative behavior. During recent years, Shadowserver.org, Spamhaus, and other individuals/organizations have exercised takeovers of botnet infrastructure or botnet-employed domains. Such achievements can have the significant side effect of perturbing empirical measurement: takedowns introduce "unnatural" activity in collected datasets [23].

To assess the magnitude of these issues, we analyzed which of the 10,670 executed samples showed network activity, but apparently failed to bootstrap malicious activities. We used Avira to identify the malware families. Only 4,235 (39.7%) of the 10,670 samples showed any network activity. Of these samples, we found that of the 22 families with at least 5 distinct samples showing any HTTP activity, 14 (63%) included samples that only had failing HTTP communication (HTTP response codes 4XX/5XX). Similarly, of the most prevalent 33 families that used DNS, eight (24%) contained samples that did not have any other communication than the (typically failed) DNS lookups. We observed such inactive samples to be more prevalent in some families (e.g., Hupigon 85%, Buzus 75%), while other families (e.g., Allaple 0%) were less affected.

Next, we tried to quantify the effects of sinkholed malware infrastructure. We contacted various security organizations to obtain information about sinkholed C&C servers. These contacts enabled us to obtain sinkholing patterns of four different organizations operating sinkholing infrastructure. We then searched for these patterns for sinkholed samples among the 4,235 samples showing network activity. Most significantly, we found that during 59 of the 394 Sality executions (15%) and 27 of the 548 Virut executions (5%), at least one sinkholed domain was contacted. Although we are aware of additional malware families in our dataset to have sinkholed domains (e.g., Renos/Artro, Gozi, TDSS, SpyEye, ZeuS, Carperb, Vobfus/Changeup, Ramnit, Cycbot), we could not spot sinkholing of these in our sample dataset. Combining this data, this translates to the observation that *at least* eleven of the 126 active families (8.7%) in our dataset are potentially affected by sinkholing.

In summary, execution of inactive or sinkholed samples will not yield representative activity, highlighting the need for authors to consider and quantify their impact.

E. Artifacts

Due to the specific setups of malware execution environments, the artifacts introduced into recorded malware traces can be manifold. For example, network traffic contains specific values such as the environment's IP address or the Windows user name. We found such easy-to-spot artifacts widespread across many malware families. Specifically, we analyzed which of the recorded network traces contain the contained environment's IP address, Windows user name, or OS version. For instance, more than 10% of all Virut samples that we executed transmitted the bot's public IP address in plaintext. Similarly, one in five Katusha samples sent the Windows user name to the C&C server. The use of "`Windows NT 5.1.2600`" as HTTP User-Agent, as for example by Swizzor (57%) or Sality (52%), likewise occurs frequently. These illustrative examples of payload artifacts are incomplete, yet already more than a third (34.7%) of the active malware families in our dataset communicated either Sandnet's external IP address, our VM's MAC address, the VM's Windows username, or the exact Windows version string in plaintext in at least one case.

More dangerous types of biases may hide in such datasets, unbeknownst to researchers. For instance, methodologies relying on time-based features should consider artifacts introduced by specific network configurations, such as limited bandwidth during malware execution. Similarly, containment policies may bias the analysis results. For example, we have observed spambots that cease running if a containment policy redirects their spam delivery to a local server that simply accepts all incoming mail.

In general, it is hard to measure the exact ratio of malware families generating any artifact. Some artifacts, such as limited bandwidth or particular system configurations such as installed software, are inherent to all malware families. Consequently, authors need to carefully consider artifacts for each experiment. The best advice to preclude artifacts is to either carefully and manually assemble a dataset, or to perform representative real-world experiments.

VI. Related Work

Prudent experiments: Kurkowski et al.'s survey [29] of the technical quality of publications in the Mobile Ad Hoc Networking community inspired our methodology. As their survey's verification strategies do not immediately apply to our community's work, we needed to establish our own review criteria. Krishnamurthy and Willinger [28] have identified common methodological pitfalls in a similar fashion to ours, but regarding the Internet measurement community. They established a set of standard questions authors ought to consider, and illustrate their applicability in a number of measurement scenarios. Closer to our community, Aviv and Haeberlen have discussed a set of challenges in evaluating botnet detectors in trace-driven settings [1], and proposed distributed platforms such as PlanetLab as a potential enabler for more collaborative experimentation and evaluation in this space. Moreover, Li et al. [31] explored difficulties in evaluating malware clustering approaches. Supporting our observations, they observed that using balanced and well-designed datasets have significant effects on evaluation results. They then show the importance of creating ground truths in malware datasets, broaching concerns related to some guidelines in this paper.

Perhaps most closely related to our effort is Sommer and Paxson's approach to explaining the gap between success in academia and actual deployments of anomaly-based intrusion detection systems [43]. The authors find five reasons: (1) a very high cost of errors; (2) lack of training data; (3) a semantic gap between results and their operational interpretation; (4) enormous variability in input data; and (5) fundamental difficulties for conducting prudent evaluations. In fact, the anomaly detection community has suffered from these problems for decades, whereas experiments with malware datasets are increasingly used in our community. Consequently, our work complements theirs in that we shift the focus from anomaly detection to malware experiments in general.

Dynamic analysis evasion: Malware datasets typically stem from dynamic analysis in specially prepared environments [8, 16, 22, 36, 42, 50]. To ensure diverse datasets, malware must not evade dynamic analysis. Others have studied the extent to which malware can detect and evade dynamic analysis [14, 32, 37]. Chen et al. present a taxonomy of dynamic analysis fingerprinting methods and perform an analysis to which extend these are used [14]. Paleari et al. present methods to automatically generate tests that effectively detect a variety of CPU emulators [37]. Most recently, Lindorfer et al. [32] analyzed how and to which extent malware samples evade Anubis.

Survey on malware detection systems: Stinson and Mitchell [45] presented a first approach to evaluate existing botnet detection methodologies. They focus on possible evasion methods by evaluating six specific botnet detection methodologies. Their survey is orthogonal to ours, as we explore how authors design experiments with malware datasets. Further, we provide guidelines how to define prudent experiments that evaluate methodologies in absence of any evasion techniques. In addition, we assist researchers in designing experiments in general rather than evaluating specific methodologies.

VII. Conclusion and Discussion

In this work we have devised guidelines to aid with designing prudent malware-based experiments. We assessed these guidelines by surveying 36 papers from our field. Our survey identified shortcomings in most papers from both top-tier and less prominent venues. Consequently, we argue that our guidelines could have significantly improved the prudence of most of the experiments we surveyed.

But what may be the reasons for our discouraging results? The observed shortcomings in experimental evaluation likely arise from several causes. Researchers may not have developed a methodical approach for presenting their experiments, or may not see the importance of detailing various aspects of the setup. Deadline pressures may lead to a focus on presenting novel technical content as opposed to the broader evaluation context. Similarly, detailed analyses of experimental results are often not given sufficient emphasis. In addition, page-length limits might hamper the introduction of important aspects in final copies. Finally, researchers may simply overlook some of the presented hidden pitfalls of using malware datasets.

Many of these issues can be addressed through devoting more effort to presentation, as our transparency guidelines suggest. Improving the correctness and realism of experiments is harder than it seems, though. For instance, while real-world scenarios are vital for realistic experiments, conducting such experiments can prove time-consuming and may raise significant privacy concerns for system or network administrators. Furthermore, it is not always obvious that certain practices can lead to incorrect datasets or lead to unrealistic scenarios. For example, it requires great caution to carefully think of artifacts introduced by malware execution environments, and it is hard to understand that, for example, experiments on overlay datasets may be biased. The significance of imprudent experiments becomes even more important in those instances where current practices inspire others to perform similar experiments—a phenomenon we observed in our survey.

We hope that the guidelines framed in this paper improve this situation by helping to establish a common set of criteria that can ensure prudent future experimentation with malware datasets. While many of our guidelines are not new, we witnessed possible improvements to experiments for every one of the criteria. We believe this approach holds promise both for authors, by providing a methodical means to contemplate the prudence and transparent description of their malware experiments, and for readers/reviewers, by providing more information by which to understand and assess such experiments.

VIII. Acknowledgments

We thank our shepherd David Brumley for his support in finalizing this paper. We also thank all anonymous reviewers for their insightful comments. We thank all our anonymous malware sample feeds. Moreover, we thank Robin Sommer for his valuable discussion input. This work was supported by the Federal Ministry of Education and Research of Germany (Grant 01BY1110, MoBE), the EU "iCode" project (funded by the Prevention, Preparedness and Consequence Management of Terrorism and other Security-related Risks Programme of the European Commission DG for Home Affairs), the EU FP7-ICT-257007 SysSec project, the US National Science Foundation (Grant 0433702) and Office of Naval Research (Grant 20091976).

References

[1] A. J. Aviv and A. Haeberlen. Challenges in experimenting with botnet detection systems. In *USENIX 4th CSET Workshop, San Francisco, CA*, August 2011.

[2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated Classification and Analysis of Internet Malware. In *10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2007.

[3] D. Balzarotti, M. Cova, C. Karlberger, C. Kruegel, E. Kirda, and G. Vigna. Efficient Detection of Split Personalities in Malware. In *17th Annual Network and Distributed Systems Security Symposium (NDSS), San Diego, CA*, February 2010.

[4] P. Barford and M. Blodgett. Toward Botnet Mesocosms. In *USENIX 1st Workshop on Hot Topics in Understanding Botnets (HotBots), Cambridge, MA*, April 2007.

[5] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, Behavior-Based Malware Clustering. In *16th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA*, February 2009.

[6] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel. A View on Current Malware Behaviors. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), Boston, MA*, April 2009.

[7] U. Bayer, E. Kirda, and C. Kruegel. Improving the Efficiency of Dynamic Malware Analysis. In *25th ACM Symposium On Applied Computing (SAC), Sierre, Switzerland*, March 2010.

[8] U. Bayer, C. Kruegel, and E. Kirda. TTAnalyze: A Tool for Analyzing Malware. In *16th Annual EICAR Conference, Hamburg, Germany*, April 2006.

[9] Bit9. FileAdvisor. http://fileadvisor.bit9.com.

[10] B. Bowen, P. Prabhu, V. Kemerlis, S. Sidiroglou, A. Keromytis, and S. Stolfo. BotSwindler: Tamper Resistant Injection of Believable Decoys in VM-Based Hosts for Crimeware Detection. In *13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2010.

[11] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *20th USENIX Security Symposium*, August 2011.

[12] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: Enabling Active Botnet Infiltration using Automatic Protocol Reverse-Engineering. In *ACM CCS*, 2009.

[13] Z. B. Celik, J. Raghuram, G. Kesidis, and D. J. Miller. Salting Public Traces with Attack Traffic to Test Flow Classifiers. In *USENIX 4th CSET Workshop*, August 2011.

[14] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario. Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware. In *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Achorage, AK*, June 2008.

[15] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol Specification Extraction. *IEEE S&P*, 2009.

[16] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. In *15th ACM Computer and Communications Security Conference (CCS), Alexandria, VA*, October 2008.

[17] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *17th USENIX Security Symposium, San Jose, CA*, August 2008.

[18] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *USENIX Security Symposium*, 2007.

[19] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *16th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA*, February 2008.

[20] G. Jacob, R. Hund, C. Kruegel, and T. Holz. JACKSTRAWS: Picking Command and Control Connections from Bot Traffic. In *20th USENIX Security Symposium*, August 2011.

[21] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *18th ACM Conference on Computer and Communications Security (CCS), Chicago, IL*, pages 309–320, October 2011.

[22] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying Spamming Botnets Using Botlab. In *6th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA*, April 2009.

[23] C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, and S. Savage. The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. In *USENIX LEET*, 2008.

[24] E. Kirda, C. Kruegel, G. Banks, and G. Vigna. Behavior-based Spyware Detection. In *15th USENIX Security Symposium, Vancouver, Canada*, August 2006.

[25] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and Efficient Malware Detection at the End Host. In *USENIX Security Symp.*, 2009.

[26] C. Kolbitsch, T. Holz, C. Kruegel, and E. Kirda. Inspector Gadget: Automated Extraction of Proprietary Gadgets from Malware Binaries. In *30th IEEE Symposium on Security & Privacy*, New York, USA, May 2009.

[27] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. GQ: Practical Containment for Measuring Modern Malware Systems. In *ACM Internet Measurement Conference*, 2011.

[28] B. Krishnamurthy and W. Willinger. What are our standards for validation of measurement-based networking research? In *ACM SIGMETRICS*, June 2008.

[29] S. Kurkowski, T. Camp, and M. Colagrosso. MANET Simulation Studies: The Incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9:50–61, October 2005.

[30] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christoderescu, and E. Kirda. AccessMiner: Using System-Centric Models for Malware Protection. In *ACM CCS*, October 2010.

[31] P. Li, L. Liu, D. Gao, and M. K. Reiter. On Challenges in Evaluating Malware Clustering. In *13th International Symposium on Recent Advances in Intrusion Detection*, 2010.

[32] M. Lindorfer, C. Kolbitsch, and P. Comparetti. Detecting Environment-Sensitive Malware. In *14th International Symposium on Recent Advances in Intrusion Detection*, 2011.

[33] L. Liu, S. Chen, G. Yan, and Z. Zhang. BotTracer: Execution-based Bot-like Malware Detection. In *11th Information Security Conference (ISC), Teipei, Taiwan*, September 2008.

[34] C. Livadas, B. Walsh, D. Lapsley, and T. Strayer. Using Machine Learning Techniques to Identify Botnet Traffic. In *IEEE LCN*, November 2006.

[35] J. A. Morales, A. Al-bataineh, S. Xu, and R. Sandhu. Analyzing and Exploiting Network Behaviors of Malware. In *SecureComm*, 2010.

[36] Norman ASA. Norman Sandbox. http://www.norman.com/security_center/security_tools/.

[37] R. Paleari, L. Martignoni, G. Fresi Roglia, and D. Bruschi. A Fistful of Red-Pills: How to Automatically Generate Procedures to Detect CPU Emulators. In *USENIX WOOT*, 2009.

[38] R. Perdisci, W. Lee, and N. Feamster. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *USENIX NSDI*, 2010.

[39] K. Rieck, T. Holz, C. Willems, P. Duessel, and P. Laskov. Learning and Classification of Malware Behavior. In *DIMVA*, July 2008.

[40] K. Rieck, G. Schwenk, T. Limmer, T. Holz, and P. Laskov. Botzilla: Detecting the Phoning Home of Malicious Software. In *25th ACM Symposium on Applied Computing (SAC)*, 2010.

[41] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic Analysis of Malware Behavior using Machine Learning. In *Journal of Computer Security*, 2011.

[42] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. C. Freiling, and N. Pohlmann. Sandnet: Network Traffic Analysis of Malicious Software. In *ACM EuroSys BADGERS*, February 2011.

[43] R. Sommer and V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *31st IEEE Symposium on Security & Privacy*, May 2010.

[44] E. Stinson and J. C. Mitchell. Characterizing Bots Remote Control Behavior. *DIMVA*, 2007.

[45] Stinson, Elizabeth, and Mitchell, John C. Towards Systematic Evaluation of the Evadability of Bot / Botnet Detection Methods. In *USENIX WOOT*, July 2008.

[46] B. Stock, M. Engelberth, F. C. Freiling, and T. Holz. Walowdac Analysis of a Peer-to-Peer Botnet. In *European Conference on Computer Network Defense (EC2ND)*, 2009.

[47] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. FIRE: FInding Rogue nEtworks. In *25th Annual Computer Security Applications Conference (ACSAC)*, 2009.

[48] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. Botnet Detection Based on Network Behavior. *Advances in Information Security, 2008*, 36:1–24, 2008.

[49] The Shadowserver Foundation. bin-test. http://bin-test.shadowserver.org/.

[50] C. Willems, T. Holz, and F. Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. In *31st IEEE S&P Magazine*, pages 32–39, March 2007.

[51] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically Generating Models for Botnet Detection. In *ESORICS*, September 2009.

[52] T.-f. Yen and M. K. Reiter. Traffic Aggregation for Malware Detection. In *5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, July 2008.

[53] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, and Y. Zhao. Online Botnet Detection Based on Incremental Discrete Fourier Transform. *Journal of Networks*, 5(5):568–576, 2010.

[54] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster. Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling. *6th ACM ASIACCS*, 2011.

[55] Z. Zhu, V. Yegneswaran, and Y. Chen. Using Failure Information Analysis to Detect Enterprise Zombies. In *SecureComm*, September 2009.