Experimental Challenges in Cyber Security: A Story of Provenance and Lineage for Malware

Tudor Dumitraș
Symantec Research Labs
tudor_dumitras@symantec.com

Iulian Neamtiu
University of California, Riverside
neamtiu@cs.ucr.edu

Abstract

Rigorous experiments and empirical studies hold the promise of empowering researchers and practitioners to develop better approaches for cyber security. For example, understanding the provenance and lineage of polymorphic malware strains can lead to new techniques for detecting and classifying unknown attacks. Unfortunately, many challenges stand in the way: the lack of sufficient field data (e.g., malware samples and contextual information about their impact in the real world), the lack of metadata about the collection process of the existing data sets, the lack of ground truth, the difficulty of developing tools and methods for rigorous data analysis.

As a first step towards rigorous experimental methods, we introduce two techniques for reconstructing the phylogenetic trees and dynamic control-flow graphs of unknown binaries, inspired from research in software evolution, bioinformatics and time series analysis. Our approach is based on the observation that the long evolution histories of open source projects provide an opportunity for creating precise models of lineage and provenance, which can be used for detecting and clustering malware as well.

As a second step, we present experimental methods that combine the use of a representative corpus of malware and contextual information (gathered from end hosts rather than from network traces or honeypots) with sound data collection and analysis techniques. While our experimental methods serve a concrete purpose—understanding lineage and provenance—they also provide a general blueprint for addressing the threats to the validity of cyber security studies.

1 Introduction

The experimental scientist carries the burden of proving that results reported to the scientific community are valid. Such proofs of validity are predicated on several conditions: the availability of relevant data sets, the integrity of data collection and analysis processes, the generalizability of results. Establishing validity is particularly challenging in cyber security because of shortcomings in current experimental methods and data sets. The technical, legal, and ethical obstacles for sharing sensitive artifacts, such as malware samples or sensitive information on networked devices compromised by cyber attacks (e.g., IP addresses or geolocation data), prevents independent verification of results. Additionally, the validation of such experimental results requires a ground truth, e.g., samples that have been labeled, objectively, as belonging to the same malware family. However, only the malware creators have access to such objective information.

We illustrate these experimental challenges by discussing the threats to validity for reconstructing the lineage and provenance of malware samples. *Provenance* refers to "documenting the entities, systems, and processes that operate on and contribute to data of interest" [20]. In this paper, provenance reconstruction means finding the compiler, development environment, testing methods, operating system, author, and geographic/network source of an attack. Effective mechanisms for detecting provenance—attributing an attack to a responsible party—act as a powerful deterrent to potential attackers. *Lineage* refers to the ancestors and descendants of an artifact. Establishing lineage allows us to piece together the timeline of an attack and helps us identify new variations of an old attack (e.g., mutations of a known polymorphic virus) [15, 14]. The result of lineage analysis is a phylogenetic tree that shows the evolution relationship between artifacts.

While lineage and provenance hold the promise of improving a variety of security tasks, from malware characterization to threat detection and cyber-attack prevention, these topics remain under active investigation. To advance this research, we propose two new techniques and we outline the requirements for their experimental evaluation. The first technique leverages open source reposito-

ries for tuning lineage reconstruction algorithms that can then be employed on malware. The second technique is based on combining dynamic analysis with recent advantages in time series analysis for classifying and detecting malware. The effectiveness of these techniques depends on the availability of training data that is representative for the current malware landscape. Moreover, a rigorous evaluation of this effectiveness must overcome the obstacles for applying the techniques to all the relevant test data (e.g., to packed malware samples) and the difficulty of generalizing the results beyond the data set used in the experiment (e.g., to malware for mobile platforms). Overcoming these threats to the validity of lineage and provenance reconstructions is challenging.

The problem of testing the effectiveness of lineage and provenance techniques embodies the general challenges of cyber security studies. We focus on *data collection and analysis* rather than on the design of a test bed for repeatable experimentation, which has been addressed in the past [7, 2, 11]. In our experience, the data collection and analysis methods are the main sources of threats to the validity of experimental results in security. Furthermore, while past research has paid considerable attention to data collected from honeypot deployments [5, 16] or from the network [18, 24, 2], we focus on *data gathered on the end-hosts* that are the targets of attacks in the real world.

In summary, our contributions are threefold:

- We propose a solution to the concrete obstacles faced by lineage and provenance research. Our approach combines techniques from machine learning and time series analysis for reconstructing malware lineage and provenance (Section 2).
- We identify the main obstacles for experimenting with lineage and provenance techniques, and we discuss several candidate approaches for addressing the threats to validity (Section 3). We also outline rigorous experimental methods for training and evaluating our proposed techniques, based on the collection and analysis of end-host data (Section 4).
- We describe the practice of reasoning about the general threats to validity in empirical studies, and we identify the challenges for overcoming these threats in the context of cyber security studies (Section 5).

2 Provenance and Lineage Reconstruction

We now present an approach for reconstructing malware provenance and lineage. Our approach uses information available from a multitude of sources (e.g., open source code repositories, malware repositories, contextual data on malware propagation) and a variety of techniques (e.g., static and dynamic analysis, context-based analysis).

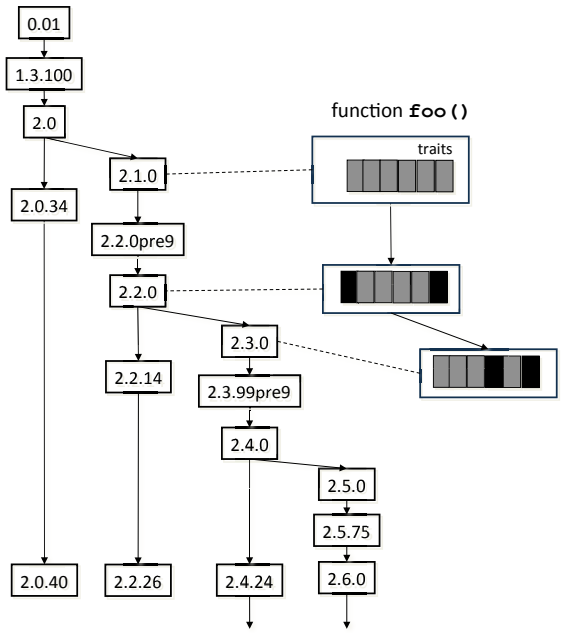

Figure 1: Partial phylogenetic tree of the Linux kernel.

Phylogenetic trees. Originating from evolutionary biology, where they show the relationships between species, phylogenetic trees are useful representations for capturing the lineage and evolution of software artifacts, both goodware and malware. Our approach is to employ phylogenetic trees constructed from the evolution of open source programs to tune lineage reconstruction techniques that can then be applied to malware.

Open source repositories provide excellent training data for modeling the evolution of binary artifacts and reconstructing their phylogenetic trees. We now have access to the source code of large, long-lived programs such as the Linux kernel, BSD operating systems, Sendmail; their release histories span hundreds of versions and 15–20 years of evolution. By simply compiling this source code, we obtain a complete phylogenetic tree of binary artifacts.

In Figure 1 (left) we present a partial phylogenetic tree of the Linux kernel, where boxes represent versions, and arrows indicate the ancestor-descendant relationship; for legibility we omit intermediate versions, e.g., between 2.0.34 and 2.0.40. By comparing the object code of the same function across multiple versions in the phylogenetic tree, we can determine (i) effective similarity measures between binary artifacts, which will help identify viruses in the same family; (ii) lineage, i.e., the direct descendants, hence the direction of the evolution; (iii)

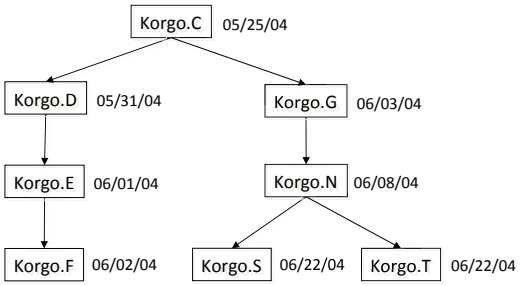

Figure 2: Partial phylogenetic tree of the Korgo worm family.

heredity, i.e., how traits are passed from ancestors to descendants. In Figure 1 (right) we show heredity information between binary artifacts by focusing on function `foo()` in versions 2.1.0, 2.2.0, and 2.3.0. We can view each binary as being characterized by a set of *traits*, which can be determined through static or dynamic analysis. Some of these traits are passed to the descendants (in gray), while some traits are not (in black). Traits serve as markers for each version in the phylogenetic tree.

Static analysis. Techniques for constructing control flow graphs, inspired from the design and implementation of compilers, provide the opportunity to identify several traits for static analysis, such as the content and order of basic blocks (i.e., sequential code without control flow statements). Other examples of static traits include dataflow information, memory access patterns, system call numbers and system call arguments. As an alternative to such semantic analysis, code n-grams can be used for capturing the sequence of instructions in the binary [14].

In the next phase, we use these classifiers to establish lineage and heredity for malware, for those cases where malware lends itself to static analysis, as follows: extract traits for known samples from certain malware families and use these as training sets for the classifier; given an unlabeled malware sample, the classifier will identify the closest family and variant for that sample.

We illustrate a malware phylogenetic tree using the Korgo worm family (named W32.Korgo.* in Symantec's nomenclature [26]). Each variant infects Windows systems by exploiting a vulnerability in the LSAS service, which represents the distiguishing feature of the Korgo family. In Figure 2 we present several variants of the Korgo worm; arrows indicate ancestor-descendant relationships, while the date next to each variant represents contextual information (first reported date, as per Symantec's list of threats and risks [26]); for example, Korgo.E, first reported on 06/01/04, is a variant of Korgo.D, first

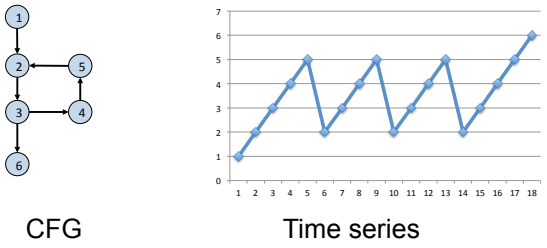

Figure 3: Representing a control flow graph as a time series.

reported on 05/31/04. In our approach, the common trait of Korgo family members is exploiting the LSAS vulnerability; we use the techniques honed on open source software to extract this common trait and establish Korgo's lineage and heredity.

Dynamic analysis. Because this static analysis approach does not work on packed malware, we describe a second technique, based on dynamic analysis. We run the unknown binary and record its dynamic control flow graph. Next, we convert this graph into a time series, where each data point corresponds to the node in the graph and the time it was observed; the amplitude of each data point corresponds to the node's topological rank in the control flow graph. We illustrate this in Figure 3; on the left we have the control-flow graph, on the right we have the corresponding time series. Each node in the control-flow graph has an associated *id* (1..6), and this *id* is used as the amplitude in the time series. On the right-hand side of the figure, we see the results, as a time series, of executing the program; the x-axis represents time, i.e., completion of a basic block, while the y-axis represents the *id* of the basic block just completed. Recent dynamic analysis work has found that different strains of malware from the same family exhibit similar dynamic control flow graphs [12]. Therefore, when translating the graphs into time series, the time series will be similar.

The last step in our approach is to reconstruct malware lineage, based on clustering and similarity metrics, e.g., by exploiting recent advances in time series analysis (motifs, search and similarity measures) [1, 6]. Techniques based on time-series similarity can also be employed for malware detection and even for preventing zero-day attacks: if the time series of an artifact of unknown origin is highly similar to the time series of known malware, the artifact could possibly be a new, previously-unknown strain of an existing virus family.

Contextual analysis. The binary samples are usually not enough for reconstructing the complete lineage and provenance of malware families. For example, malware

creators can employ obfuscation techniques for packing the executable code and for randomizing the system call sequences, which reduces the accuracy of static and dynamic analysis techniques. Additional contextual information is needed to understand when each variant has appeared in the wild and how it has propagated from host to host. Such information can be derived from field data, e.g., network traces or infection reports, collected from multiple sensors around the world, as explained in Section 4.

These sensors could provide the timestamps required to establish heredity relationships (parent-child), as similarity metrics alone cannot establish the direction of evolution (as illustrated in the Korgo example). Time-series similarity techniques are well-suited for reconstructing the timeline of a malware family's evolution.

3 Challenges for Rigorous Experimental Studies

Drawing meaningful conclusions from the results of our lineage and provenance techniques hinges on our ability to train and benchmark them rigorously, to ensure that the experimental results are accurate indicators for how well these techniques would perform in the real world. Such rigorous experimental methods place several requirements on the experimenters and the data sets they use. For example, representative field data is essential for training classifiers and conducting empirical studies, and the ground truth must be available, for validating the experimental results. Metadata describing the data collection process and the experimental procedures is also required for ensuring the repeatability of experiments and the independent verification of results. These prerequisites are difficult to achieve for experimental studies in computer security.

3.1 Candidate Approaches

A number of existing approaches make headway toward rigorous experimentation. For example, the Emulab [11] and DETER [2] test beds make it easier for researchers to ensure that their experimental processes are repeatable. However, the lack of representative field data and of a uniform benchmarking methodology continue to threaten the validity of cyber security studies. Unlike in the fields where benchmarking is well established, such as computer architecture, systems, or databases, the data sets used for validating computer-security research are often mentioned in a single publication and then forgotten. This, in itself, represents a threat to the validity of these studies, as it does not accommodate meaningful comparisons against prior work, and raises a barrier to generalizing the conclusions [22].

The security research community emphasized the current need for representative data in a "data wish list," compiled by Camp et al. [3]. In addition to specific data that is currently unavailable, the authors identify the need for a data-sharing process that facilitates the collection of metadata and that addresses privacy and legal concerns.

Synthetic data. Some benchmarking efforts in the past have addressed these concerns by generating synthetic input data—a technique widely used for assessing the performance of computer systems [8, 23]. For example, the Lincoln Laboratory evaluation of intrusion detection systems [18] uses synthetic attack and background data, generated from statistical distributions observed in network traffic from several Air Force bases. This approach provides considerable flexibility, because the data generators allow an experimenter to explore all the behavioral corner cases of the system-under-test. When assessing system security, however, synthetic data has a short-lived relevance because the cyber threat landscape changes at a fast pace.

McHugh [21] criticizes the Lincoln Laboratory data set for the lack of information on the validation of test data, such as measures of similarity with the original traffic traces. Another threat to validity lies in the difficulty to generalize the results, i.e., to conclude that similar behaviors should be expected when exposing the systems-under-test to real world data. For example, the false positive rate of intrusion detection systems is influenced by background noise, which should be consistent with the background data that the system is likely to encounter in a real deployment. McHugh observes that the experimenter has the burden of proof for showing that the artificial environment does not affect the outcome of the experiment. These lessons endure in the community: the PREDICT data repository [9] was also criticized for the lack of adequate metadata, and Camp et al. [3] emphasize the need for metadata that allows experimenters to distinguish meaningful conclusions from spurious observations.

Field data. A similar challenge of establishing representativeness applies to benchmarks based on field data, gathered from production systems; an example would be data on threats to obsolete versions of operating systems and applications. Such data collections must be updated frequently in order to account for changes in the cyber threat landscape. A more subtle challenge for a benchmark that incorporates raw field data is to allow the experimenter to control for factors that might affect the interpretation of the results. Even when the experimental results are statistically significant, owing to the large amount of data analyzed, their validity can still be questioned in cases where small changes in the

collection process (e.g., input sources or sampling algorithms) can drastically alter the outcome of the experiment. The benchmark should provide the *collection metadata* needed for establishing the real-world situations that each data set is representative of. Moreover, in order to make it possible for future projects to reproduce the experimental results, the benchmark must provide tools for recording the *experiment metadata*: the hypotheses tested and their accept/reject criteria, the experimental design (e.g., the unpacker used to pre-process the data, the static/ dynamic analysis tools employed, the classification features and their rationale), the scripts and procedures used for data analysis, the statistical apparatus employed.

Security-oriented field data must necessarily include sensitive artifacts, such as dangerous binaries (e.g., malware) and data that could reveal personally identifiable information (e.g., hosts that have been compromised by attackers). For example, several research efforts in the past have collected malware samples from honeypot deployments [5, 16] and information about the real-world behavior of malware from network traces [18, 24, 2]. Drawing conclusions from such experiments is challenging: in the first case the researchers must prove that the honyepot instrumentation did not alter the experimental results, and in the second case the network packets reveal only certain aspects of malware behavior.

A more representative image of the cyber threat landscape could be painted by malware samples and behavioral data collected on a large number of end-hosts—enterprise and consumer PCs that are the victims of cyber attacks. In the rare cases where a research project is able to collect such data, by instrumenting a representative sample of end hosts and by establishing data management agreements with their users, the resulting data set is not easy to share with other researchers. For example, the IP addresses of hosts initiating network-based attacks could point to personal computers that have been infected with malware, while the country codes of the attack destinations reveal further sensitive information. Binary samples of malware collected in this manner cannot be made publicly available. Such ethical and scientific challenges have prevented, so far, the establishment of a representative corpus of data for cyber security benchmarking and experimentation.

3.2 Problems for Lineage and Provenance

These challenges for collecting and sharing security-oriented data result in a number of fundamental threats to the validity of lineage and provenance studies. The first step in determining lineage is clustering samples into malware families. Prior approaches have used classification features such as the similarity of string signatures or of dynamic call graphs to identify viruses in the same family [12]. The inability of current tools to unpack obfuscated code or to compute dynamic call graphs, which makes it difficult to expose the connection between independent variables, e.g., the traits chosen as classification features, and dependent variables, e.g., malware families.

The second step in reconstructing the lineage involves assembling a phylogenetic tree. This tree indicates, for example, not only that Korgo.E and Korgo.F are mutations of Korgo.D, but also that Korgo.D precedes Korgo.E, which precedes Korgo.F. Tree construction depends on comparing the distance between classification features and on using contextual information (e.g., time when a virus was first reported, semantics reconstructed by dynamic analysis and expert analysis). The lack of relevant information from malware factories, on their development processes and dissemination strategies, prevents us from establishing a ground truth for validating lineage results.

Similarly, provenance techniques aim to use the executable information to reconstruct the development environment (e.g., compiler or IDE) used to produce malware. For example, different C++ compilers use different virtual-method table layouts, and, similarly, code segment layouts produced by C compilers differ as well; this further helps identify the toolset used for developing the malware. However, malware written in assembly contains no such distinguishing information, which prevents these environment-reconstruction techniques from operating on all the relevant training data. Additionally, pinpointing the geographical or network location where the attack was launched requires contextual information, collected worldwide, on the dissemination of the threat. The challenges for collecting such contextual information in a comprehensive manner [22, 10] further complicate provenance analysis.

4 Experimentation on End-Host Data

Without the ability to reproduce an experiment, it is impossible to conclude that the results generalize beyond the data set used in the experiment [17]. In other words, experimental results that cannot be verified independently lack *external validity*. Moreover, the data sets must be updated continuously to ensure that the data is relevant to the current cyber threat landscape—i.e., to ensure the *content validity* of experiments. These, as well as other threats to validity (described in more detail in Section 5), are often caused by the data collection process and the experimental methods employed [22].

For lineage and provenance, collecting and studying end-host data facilitates experiments that can yield general and representative results. For example, malware samples and contextual information collected on end

Data set	Sources
Malware samples	200 countries
Binary reputation	50 million machines
Anti-virus telemetry	130 million machines

Table 1: Data sets included in WINE [10].

hosts around the world provide a diverse data set for experimentation, likely to cover a large number of malware families. Reconstructing the timeline of cyber attacks, for lineage and provenance studies, can also benefit from observations made on end hosts, because network traces may not reveal which member of a malware family was released first (e.g, different members of the same malware family may produce similar network traffic) and because the malicious behavior may not be reproducible in the lab (e.g., bots usually become inoperative after their command and control nodes have been neutralized). This example further emphasizes the need to update the corpus of field data continuously, in order to reflect the frequent changes in the cyber threat landscape.

Updating the data sets in this manner is in apparent conflict with the reproducibility requirement. Fortunately, this conflict is superficial. We should focus on defining a predictable process for data collection, while preserving the reference data sets employed in prior experiments. This would facilitate experiments that produce representative results, based on the freshest data, and that also remain reproducible in the future.

An example of a field-gathered data corpus, suitable for lineage and provenance studies, is the WINE benchmark, curated by Symantec Research Labs [10]. The WINE data sets include malware samples, binary reputation and anti-virus telemetry (see Table 1). The malware collection includes representative samples of both packed and unpacked malware (e.g., viruses, worms, bots), which are used for creating Symantec's anti-virus signatures. The binary reputation data provide information on unknown binaries (i.e., files for which an anti-virus signature has not yet been created) that are downloaded by users who opt in for Symantec's reputation-based security program. The anti-virus telemetry records occurrences of known threats, for which Symantec has created signatures and which are detected by anti-virus products. The WINE data sets also specify the collection metadata, specifying when, where and how the information was recorded.

The malware data set provides a large number of variations of each malware family, collected around the world, which improves the accuracy of lineage reconstructions and ensures that results are representative of real-world attacks. Furthermore, the contextual information included in the binary reputation and anti-virus telemetry data can reveal when a particular malware strain first appeared, and what effect did the release of security patches and anti-virus signatures have on the production of additional variants from the same family. The order in which the various strains have appeared in the wild is relevant for lineage reconstruction, while their dissemination patterns are relevant for provenance reconstruction.

The measures of similarity between time series, described in Section 2, can potentially unlock additional secrets of this contextual information. By constructing time series of binary download rates, as recorded in WINE's binary reputation data set, we can identify patterns that are unique to certain types of malware. For example, while security patches pushed by software vendors are likely to trigger similar behavioral alarms as malware that tries to subvert the operating system's protections, the dissemination mechanisms of patches and malware are different. Similarly, malware that spreads through an aggressive spam campaign is likely to exhibit a higher download rate than legitimate software products, which rely on traditional marketing approaches.

To ensure the reproducibility of these experiments, we must preserve both the data sets employed and metadata on the experimental procedure. However, simply archiving the input data and then logging the steps taken during the experimental procedure (e.g., by recording the interactive terminal sessions) is not sufficient for achieving reproducibility. We can fill this this gap by maintaining a *lab book*[1] that documents the experimental hypothesis and the purpose of each procedural step.

Threats to validity. The techniques described in Section 2 can benefit from tuning on open source software and from training and validation using representative end-host data. Including malware families with many variants ensures that the results are representative of the technique's performance on real-world malware.

Generalizing the conclusions of such a study is more challenging. For example, we hypothesize that our lineage analysis techniques can reconstruct the lineage of JavaScript and Flash malware, which often propagates through social networks. However, we are currently unable to validate this hypothesis because the data sets currently available—including WINE—lack representative field data on attacks against social networks. Collecting such data on a large scale requires establishing many decoy accounts on the social networks, as well as accessing the system logs from their data centers. For example, the spam blacklist compiled by a social network can provide the ground truth for malware detection studies.

[1] Keeping a lab book is common practice in other experimental fields, such as applied physics or cell biology.

Similarly, we lack representative data on mobile platforms, such as iOS or Android. It is challenging to develop a data collection infrastructure for these platforms; for example, we must take into account the limited computational and input capabilities of mobile devices and the costs of network bandwidth.

5 Discussion

The problem of testing the effectiveness of lineage and provenance techniques embodies the general challenges of cyber security studies. The conclusions drawn from any empirical study are subject to multiple threats to validity: construct validity, content validity, internal validity, and external validity [25, 13]. For studies in cyber security, these threats are difficult to overcome, owing to the unique properties of security data sets and analysis methods. In this section we review the high-level requirements for ensuring the validity of cyber security experiments.

Construct validity requires using metrics and measures that actually model the hypotheses. For example, when measuring software complexity via program analysis, we know how to pick a good metric from a bad one: the number of distinct paths through a function [19] ensures construct validity, while the number of comments in that function does not. Choosing metrics for security, however, is challenging. Consider, for example, the problem of determining whether a binary of unknown provenance is malicious or not. As malware creators overwhelmingly employ packing techniques to obfuscate their code, complexity-based metrics are only as good as our ability to unpack to obfuscated code before conducting static analysis. We could overcome this challenge through dynamic analysis techniques, which execute the unknown binary in order to record its behavior. The binary must be executed in a sandboxed environment (e.g., a virtual machine) to avoid the risk of infecting the test bed and of spreading to other hosts. However, attackers increasingly adopt anti-virtualization techniques [4], which cause the virus to exhibit a different behavior when running inside a virtual machine and further thwart detection efforts. Existing program analysis techniques are therefore relevant for only a subset of today's malware, and the size of this subset is difficult to assess without field data from production systems.

Content validity requires excluding irrelevant data points and including all the relevant ones. For instance, including partially-unpacked malware samples among inputs would lead to erroneous conclusions about the detection rate of an analysis tool. Conversely, recent zero-day attacks are usually excluded from studies of the malware landscape, but they should not be. Moreover, for training and validating malware detection tools, we must also include goodware samples among the inputs. Unfortunately, a well-defined partition between malware and goodware (i.e., a ground truth) is not likely to be available for cyber-security studies, except in small-scale red-teaming experiments. Today, most binaries suspected of carrying malicious payloads are convicted through automated techniques, without involving human analysts, and using this data as a point of reference for testing new algorithms confounds the interpretation of false positive and false negative rates [17]. Moreover, as the cyber threat landscape changes at a fast pace, the conclusions based on yesterday's data are not necessarily relevant for today's attacks, hence the lack of continuously-updated data sets [3, 22] undermines content validity for security studies.

Internal validity requires that a study be properly constructed, such that a causal connection can be established between independent and dependent variables. For example, when analyzing unknown binaries on an infected system, the analysis is tainted, and, as a result, we might incorrectly classify goodware as malware. This is often a challenge for studying end-host data, because the data collection process must ensure that the host instrumentation is tamper resistant. It is the duty of the experimenter to control for variables that might influence the interpretation of the result and to ensure that changes in dependent variables can only be caused by changes in independent variables. When measuring system performance, this is usually achieved by using synthetic data generators [8, 23], which can be tuned to vary a single input variable at a time. However, synthetic data generators have a limited relevance for security (see Section 3.1), which undermines the internal validity of studies constructed in this manner [21].

External validity requires that the results of a study generalize to systems and data sets outside the scope of the study. External validity is gravely threatened in the realm of security: studies on old data do not generalize to current data; studies on certain malware types, e.g., trojans do not generalize to polymorphic viruses; and studies on network-based attacks do not generalize to host-based attacks. Moreover, as the malware has spread to social networks and smartphones, the scarcity of field data in these domains further undermines the external validity of studies based on the existing security-oriented data sets.

6 Conclusions

In this paper, we present the common threats to the validity of cyber security experiments and evaluations, and the challenges for overcoming them. For example, the obfuscation and anti-virtualization techniques frequently employed by malware creators often prevent analysis, which undermines the construct validity of security-oriented studies. The lack of ground truth about real-world malware threatens the studies' content validity. Frequent changes in the malware landscape reduce the relevance of evaluations based on synthetic data, affecting external validity, while field data makes it difficult to control for all the variables that could influence the interpretation of the result, threatening internal validity.

To ground these methodological challenges in a concrete problem, we present new techniques for reconstructing malware lineage and provenance, and the challenges for validating these techniques experimentally. We emphasize the advantages of analyzing field data, collected on end hosts around the world, for addressing the threats to the validity of lineage and provenance reconstruction.

Acknowledgments.

This research was supported by DARPA Contract FA8750-10-C-0160 sponsored by the Air Force Research Lab. We thank Marc Dacier, our shepherd Elie Bursztein, and the anonymous reviewers for their feedback on early drafts of this paper.

References

[1] AGRAWAL, R., LIN, K.-I., SAWHNEY, H. S., AND SHIM, K. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1995), VLDB '95, Morgan Kaufmann Publishers Inc., pp. 490–501.

[2] BENZEL, T., BRADEN, R., KIM, D., AND NEUMAN, C. Experiences with DETER: A testbed for security research. In *IEEE Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities* (2006).

[3] CAMP, J., CRANOR, L., FEAMSTER, N., FEIGENBAUM, J., FORREST, S., KOTZ, D., LEE, W., LINCOLN, P., PAXSON, V., REITER, M., RIVEST, R., SANDERS, W., SAVAGE, S., SMITH, S., SPAFFORD, E., AND STOLFO, S. Data for cybersecurity research: Process and "wish list". `http://www.gtisc.gatech.edu/files_nsf10/data-wishlist.pdf`, Jun 2009.

[4] CHEN, X., ANDERSEN, J., MAO, Z. M., BAILEY, M., AND NAZARIO, J. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *DSN* (2008), pp. 177–186.

[5] CHESWICK, W. R. An evening with Berferd, in which a cracker is lured, endured, and studied. In *Proceedings of the Winter USENIX Conference* (San Francisco, CA, Jan 1992).

[6] CHIU, B., KEOGH, E., AND LONARDI, S. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2003), KDD '03, ACM, pp. 493–498.

[7] DEBAR, H., DACIER, M., WESPI, A., AND LAMPART, S. An experimentation wokbench for intrusion detection systems. Tech. Rep. Research Report RZ 2998, IBM Research, Zürich, Switzerland, 1998.

[8] DEWITT, D. J. The Wisconsin benchmark: Past, present, and future. In *The Benchmark Handbook for Database and Transaction Systems*, J. Gray, Ed. Morgan Kaufmann, 1993.

[9] DHS. PREDICT, 2011. `http://www.predict.org/`.

[10] DUMITRAŞ, T., AND SHOU, D. Toward a standard benchmark for computer security research: The Worldwide Intelligence Network Environment (WINE). In *EuroSys Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* (Salzburg, Austria, Apr 2011).

[11] EIDE, E., STOLLER, L., AND LEPREAU, J. An experimentation workbench for replayable networking research. In *USENIX Symposium on Networked Systems Design and Implementation* (Cambridge, MA, April 2007).

[12] FENG, M., AND GUPTA, R. Detecting virus mutations via dynamic matching. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on* (2009), pp. 105–114.

[13] IZURIETA, C., AND BIEMAN, J. The evolution of FreeBSD and Linux. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering* (2006), pp. 204–211.

[14] JANG, J., BRUMLEY, D., AND VENKATARAMAN, S. Bitshred: Fast, scalable malware triage. Tech. Rep. CMU-CyLab-10-022, Carnegie Mellon University, 2010.

[15] KARIM, M., WALENSTEIN, A., LAKHOTIA, A., AND PARIDA, L. Malware phylogeny generation using permutations of code. *Journal in Computer Virology 1* (2005), 13–23. 10.1007/s11416-005-0002-9.

[16] LEITA, C., BAYER, U., AND KIRDA, E. Exploiting diverse observation perspectives to get insights on the malware landscape. In *International Conference on Dependable Systems and Networks* (Chicago, IL, Jun 2010), pp. 393–402.

[17] LI, P., LIU, L., GAO, D., AND REITER, M. K. On challenges in evaluating malware clustering. In *International Conference on Recent Advances in Intrusion Detection* (Ottawa, Canada, 2010), pp. 238–255.

[18] LIPPMANN, R. P., FRIED, D. J., GRAF, I., HAINES, J. W., KENDALL, K. R., MCCLUNG, D., WEBER, D., WEBSTER, S. E., WYSCHOGROD, D., CUNNINGHAM, R. K., AND ZISSMAN, M. A. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. *DARPA Information Survivability Conference and Exposition,* (2000), 12–26.

[19] MCCABE, T. J. A complexity measure. *IEEE Trans. Software Eng. 2*, 4 (1976), 308–320.

[20] MCDANIEL, P. Data provenance and security. *IEEE Security and Privacy 9*, 2 (2011), 83–85.

[21] MCHUGH, J. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security 3*, 4 (2000), 262–294.

[22] MCHUGH, J., BAYUK, J., GUPTA, M., AND MAXION, R. Science of security experimentation. Panel at 2nd Workshop on Cyber Security Experimentation and Test, Aug 2009.

[23] MENASCÉ, D. TPC-W: A benchmark for e-commerce. *IEEE Internet Computing 6*, 3 (May/Jun 2002), 83–87.

[24] PAXSON, V. Strategies for sound internet measurement. In *Internet Measurement Conference* (Taormina, Italy, Oct 2004), pp. 263–271.

[25] PERRY, D. E., PORTER, A. A., AND VOTTA, L. G. Empirical studies of software engineering: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering* (New York, NY, USA, 2000), ACM, pp. 345–355.

[26] SYMANTEC. A - Z list of all Threats and Risks, 2011. `http://www.symantec.com/security_response/threatexplorer/azlisting.jsp?azid=W`.