



Can Malware Be Exterminated by Better Understanding Its Roots?

Michael Cheng Yi Cho, Chia-Wei Hsu, Shihpyng Shieh, and Chi-Wei Wang,
National Chaio Tung University, Taiwan

Can malware be exterminated? Pessimists believe that complete malware detection is an unsolvable and nonboundable problem; optimists argue for eventual solvability. Here, the authors reveal pitfalls in malware research that, if addressed, could help move us in the right direction.

A malware-free world is currently out of reach: whatever your defense strategy, there's a counterattack example.¹ Researchers studying malware detection are actively developing new defensive approaches, yet reported security incidents continue (such as Heartbleed; <http://heartbleed.com>). Our own investigation into malware revealed various pitfalls in current research. If we could avoid such pitfalls, we might discover the path to some form of malware freedom, even if malware recognition is an undecidable problem.²

Here, we enumerate the pitfalls, and although we don't directly define how to obtain a malware-free utopia, we propose a path forward to better address the malware problem.

Malware in Cyberspace

Malware is malicious software that breaks security policies. Acts of security policy infringement include scanning, jamming, eavesdropping, spamming, and trespassing. Evidence of security policy infringement is crucial for researchers who wish to engage in malware-detection research.

Computer security policies can be mandatory or discretionary. A mandatory policy is set by organizations and enforced using automation, whereas a discretionary policy is determined by individual users. Security policy infringements can be evidenced by malware-detection engines in real time or by digital forensics using logged data.

In searching for the root of malware, the key is finding this evidence of security policy

infringement. Malware is a subcategory of software; a piece of software becomes malware when a certain security policy—whether mandatory or discretionary—has been breached. Security policies vary spatially (based on application users) or temporally (based on technology breakthroughs)—in other words, different users need different security policies at different times.

In general, the foundation of security policy violation results in confidentiality, integrity, and availability (CIA) infringement. Some argue that more attributes should be included in information security,³ but to simplify the problem, we assume CIA covers most information security aspects. Because malware violates a specific set of security policies, discovering policy infringement is crucial for malware detection; it differentiates the benign software from the malware.

Here, we focus on the *soundness* and *completeness* of malware detection. For brevity, we don't discuss the technical details of our implementation, and we make two assumptions to keep the problem bound. First, the security policies are robust and can cover most aspects of information security concerns. Second, malware execution leaves a trace in the host computer system that can be reproduced in the aftermath of an attack. These traces should provide sufficient evidence to verify security policy infringement.

Pitfalls in Malware Detection

Despite progress in information, software, and computer security research, there are pitfalls stemming from inflated assumptions about malware detection and analysis. These factors can hinder successful detection results or induce negative results. The pitfalls include

- unfairness in detection accuracy,
- unverifiable research claims,
- malware population explosion, and
- detection result disputes.

The first three deal with the need to improve malware detection research, and the fourth pitfall deals with the fact that such improvements rely on concrete evidence.

Unfairness in Detection Accuracy

The accuracy and measurement of malware research is often evaluated by false positives and

false negatives, with the measurement normally covered by two factors—soundness (eliminating false positives) and completeness (eliminating false negatives). The scale of the accuracy measurement relies on experiments conducted using large numbers of malware samples. However, most available malware samples are outdated, and collecting more recent ones for use as samples is extremely time-consuming and difficult.

Test case collection itself is a major task. Without a common, up-to-date set of test cases, accuracy measurements can be unfair due to inconsistencies in the samples. Such inconsistency is caused by the rapid evolution of malware and the unavailability of a central malware repository. Therefore, a common benchmark or dataset should be available for researchers to compare the accuracy of their research work against former efforts. However, due to the fast growing number of evolved and mutated malware programs, it's extremely difficult, if not impossible, to keep a malware test suite up to date. Furthermore, any dataset must not be a catalyst that inspires new versions of malware.

Unverifiable Research Claims

Proving or validating the claims of published research results is difficult: an exact, reproduced experiment environment is hard to mimic, and samples are hard to obtain.⁴ Examining a research solution with new samples is another problem. If research experiments are repeatable, they serve as evidence for correct detection.

Malware Population Explosion

Malware research seeks methodologies that provide accurate detection. However, the growth rate of general-purpose software has increased dramatically.⁵ To keep up, the efficiency of detecting malicious behavior quickly becomes important. If the detection speed can't at least match the malware growth rate, the number of undetected malware programs will multiply, leaving much undetected malware in operation.

Detection Result Disputes

Different detection engines might disagree concerning whether an executable is malicious. To address this, concrete evidence for a security

policy infringement should be supplied, along with all detection results. If there's a dispute, the independently gathered evidence can be used in further investigations.

The key is leveraging security policy infringement evidence that's presented in an easy-to-understand format. Although this isn't easy, we approach the issue using currently available technology.

Malware Research Idealization

Figure 1 exemplifies the relationship among malware security policies, evidence, and malware detection research.

The rectangle object in Figure 1 represents all the collected executable samples, including benign and malicious software. The three circle objects represent the three security policies—namely, CIA policies that cover all malware within the collected samples. The oval object represents malware detection research targeting a specific security property. For example, “Research A” and “Research B” target the registry modification property, while “Research C” aims for the information leakage property. Regardless of the malware detection methodology, false positives and false negatives exist.

From the perspective of Research A, false positives exist that don't overlap with the integrity violation circles—that is, the green meshed area. An executable sample is regarded as a false positive if a targeted property is found, but the evidence of information security violation is missing. On the other hand, false negatives exist in Research A if another malware detection research—namely, Research B—targeting the same property disagrees with the result. The green shadowed area represents a dispute between Research A and Research B; hence, it represents false negatives from the perspective of Research A. Another example of a false negative exists in Research C, when the true malware set for information leakage is greater than the detected population (the gray shadowed area in the “confidentiality violation” circle).

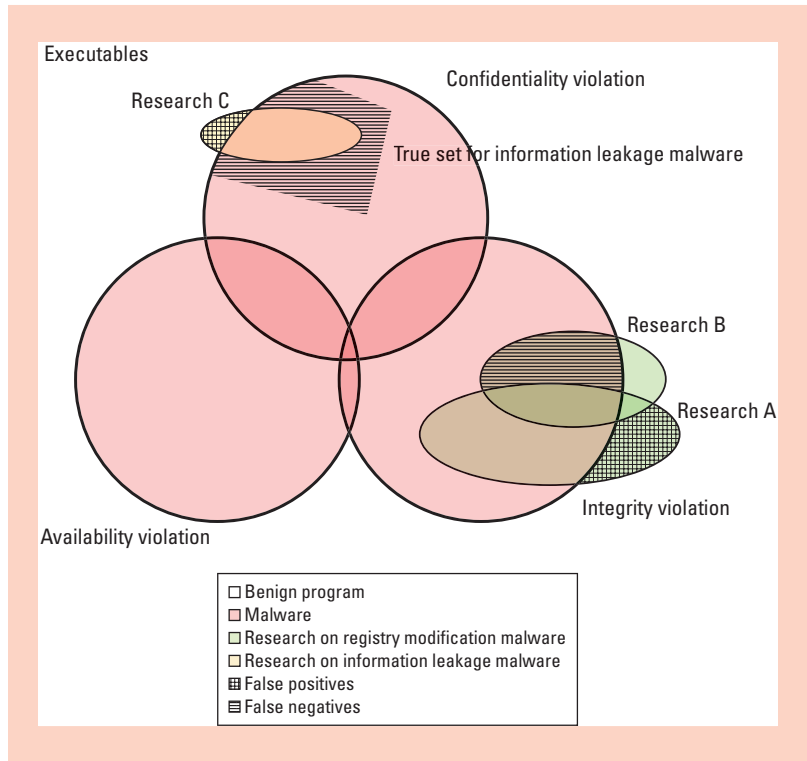


Figure 1. Malware research abstraction. The illustration shows a true malware set as well as the malware sets detected by research that attempts to approximate the true malware set by either reducing the false positives or increasing the true positives.

Figure 1 illustrates the long-term goal for malware detection research—improved accuracy. To achieve this, we exploit information security policy infringement evidence. False negatives can be eliminated using pieces of supporting evidence, which lets us list and verify the information security policies a malware sample violates. Thus, we can resolve disputes between independent malware detection results to enhance malware detection coverage and accuracy. Furthermore, the evidence can eliminate false positives: a detected suspicious behavior didn't necessarily violate a policy.

For example, it's normal for certified software in compliance with security policies to modify a Windows registry. So, evidence of an information security policy violation is a key factor in distinguishing between benign and malicious software. With security policy violation evidence, we can consolidate malware detection results and potentially eliminate false positives and negatives to predict the actual malware set. With this, we propose a preliminary approach and database design for malware set prediction.

Predicting the Malware Set

Recent research contributes to three types of improvements: new attacks,⁶⁻¹⁰ new detection methods,¹¹⁻¹⁴ and improved efficiency.^{15,16} These improvements address issues of quantity, accuracy, and efficiency.

The Cybercrime Scene Database (CSD) is a database we designed to demonstrate the feasibility of malware set approximation.

Quantity: Increasing Sample Collections

Collecting both malicious and benign executables can improve the quantity of test samples. The malicious executables can be classified into two types: newly created and newly discovered. Hackers can launch attacks using newly created malware when new attack techniques are developed. For example, the anti-virtualization malware invented in recent years can be used to hamper cloud services.

The remaining new malware programs are mutants and can be discovered using their known behaviors. Mutations include metamorphism and polymorphism that apply code obfuscation to evade detection. In contrast to malware, benign executables are evidence of information security infringement that hasn't been discovered. With a public sample collection, researchers have a common benchmark to evaluate the accuracy of their work.

Accuracy: Analyzing Fine-Grained Behavior

Malware behaviors can be analyzed at various granularities. Fine-grained malware analysis improves malware detection accuracy by increasing the chance of discovering attack evidence. Malware research results can claim higher precision rates than previous work using heuristics that usually examine samples and determine whether they perform the expected behaviors. Analysts can easily associate the expected behaviors with security policy violations. For example, smartphone adware might steal personal information and violate personal privacy protections if the International Mobile Subscriber Identity and

International Mobile Equipment Identity are sent via the Internet. These collected packets can be regarded as evidence.

The verifiable evidence can be used to convince security analysts that the adware is malicious. With fine-grained behavior analysis, a more precise behavior description can be introduced, such as contact stealing and stealth dialing. A detailed behavior examination can reduce false negatives and false positives.

Efficiency: Accelerating Malware Analysis

The study of malware analysis efficiency can speed up malware detection, thereby reducing the incubation period of new exploits. The importance of efficiency is often underestimated because most research focuses on either automation or accuracy. However, ignoring efficiency will lead to the aforementioned malware explosion problem. Efficiency can, if done properly, narrow the time needed to find new malware.

The Malware Database

The Cybercrime Scene Database (CSD) is a database we designed to demonstrate the feasibility of malware set approximation. The CSD aims to store benign and malicious code samples. Samples can be tagged with specific behaviors that violate defined security policies. Each group of samples can share a common behavior, which is regarded as one malware category, and these categories can overlap. The names of the malware categories—such as Trojan, Adware, or Rootkit—are those used by antivirus companies.

In addition to these conventional names, each category in the CSD will include a behavior description of the violation, which should be verifiable. For example, a description could be, "a program opens a backdoor without permission and sends the address book to the Internet." To standardize the user interface, the description can be formalized. Such a description could be verified by checking the created port number and captured outgoing packets. We regard this information as evidence of broken security policies because the program's behaviors are repeatable.

Ideally, all executables would be collected and verified in the CSD. The collection and verification of executables would implicitly satisfy the quantity and accuracy requirements for predicting the true malware set. This could raise the

overall malware detection rate: researchers could apply the proposed method to any system to quickly and precisely determine malware by querying its unique identifier in the CSD.

Furthermore, cloud computing offers a platform for storing test samples, searching for malware patterns, and repeating attacks. The CSD could leverage the huge storage capabilities of clouds to search for known malware patterns. Well-known scalable database systems, such as HBase, Cassandra, and Bigtable, provide opportunities here, suggesting that users might be able to insert, update, and modify data quickly. These database systems typically provide MapReduce for parallel processing to achieve large-scale search.

Finally, the virtualized environment could be used for behavior verification. The behavior of a sample could be repeated in a cloud containing detailed evidence. The behavior verification of each sample is important for dealing with malware detection. Using a reconstructable virtual environment to verify security violations would let the malware research community examine the correctness of the described security violation behavior. It's thus important for the CSD to be made publicly available to the malware research community, providing a common place to debate security violation descriptions.

This concept is similar to the current legal system, in which a prosecutor provides evidence of a law violation, and the court determines whether to convict the offender based on that evidence. Once the malware research community can agree on the security violation description of a malicious executable, we'll have a better understanding of the cause of the security violation. This would also help researchers acquire malware samples that better match the desired study description, and possibly in greater quantities, to help them pursue more accurate and efficient malware detection.

The Design

The CSD can be constructed as a 2D database in which each row represents a sample (executable) and each column represents a behavior description for security policy violations. The data indexed by a row key and a column key is one piece of evidence. The CSD is suitable for implementation as a distributed key-value database

system—specifically, NoSQL. This database is scalable for storing infinite data. Each row key is a unique identifier for a sample indexing. The unique identifier can be composed of file-related information, such as the hash value of file content, file size, and file type. Each column can be a serial number that stands for a behavior category, bounded by a behavior description. The description indicates temporary or permanent activities monitored in a machine. The change of a machine state as a result of these activities can be regarded as a piece of behavior evidence, indexed by the row and column keys.

A piece of evidence can be a snapshot image of a virtual machine (VM) or another kind of execution information that can be re-generated deterministically (however, some researchers are still studying “replay systems” that produce faithful execution in VMs; faithful execution should ensure that an execution is the same as the original execution recorded in replay systems). We store evidence as an element of the database, indexed by the row and column keys.

Based on the database's functionalities, here we introduce the properties of the CSD.

Row Insertion

To set up a common benchmark for analysts, the CSD provides row insertion for universal executable collection. Any user can upload executables; the system will then generate a unique identifier as its row key in the database. The number of rows stands for the total number of samples in the CSD, and a new executable found or created is inserted as a new row. We expect that all executables, benign or malicious, will be stored in the CSD to construct a common benchmark for analysts.

Column Insertion

The CSD will provide test sample sets, each of which will deal with a subproblem of malware detection. Although a generic solution to malware detection doesn't currently exist, solutions to subproblems with rigorous constraints might exist owing to common behaviors. These test sample sets of subproblems within the CSD will represent different malware categories. The behaviors of all malware are infinite, so letting CSD researchers insert new categories is beneficial. Each category (column) will represent a

verifiable behavior description that's observable and meaningful for security concerns. With more categories, the system could help reduce the false positives and false negatives.

Value Update

To provide provable information about security violations, the CSD offers an operation to collect the evidence of a sample. Testing all executables is difficult but possible; many automatic malware analysis methods have been proposed. We can use behavior analysis tools to perform evidence extraction for each executable.

Initially, the status of a non-tested sample is "not tested." After an extraction, the value can either be "found" or "not found." If the value is "found," analysts should upload their evidence to prove the correctness of their behavior examinations. We suggest using a VM image to achieve the property of "verifiable." With VM images, users can monitor system state changes to evaluate the accuracy of malware detection.

Query

For malware detection and research, the CSD will let users query an executable regardless of whether it contains malicious behavior. To query the CSD, a user generates the unique identifier of a file on the client side and sends the identifier as a row key to the CSD via the Internet. The CSD then reports violated security policies and malicious behaviors to users. Researchers could then perform advanced searches to collect specific sets or categories of executables. In this way, researchers could evaluate their detection method using a specific category as a common benchmark.


Behavior Verification

For credible classifications, the CSD provides repeatable evidence that can prove the presence of behaviors described in its columns, a "correctness proof" that's important for future research. With behavior verification, analysts can argue for their predictions. Evidence of an attack incurred by test samples can be as simple as a VM image. For example, an analyst's claims that a Trojan opens a port sending a string "foo bar" might be true. During execution, the monitored VM immediately produces a network packet that includes the string "foo bar." Although

sending out a "foo bar" network packet might not be harmful to most users, at least concrete evidence of "foo bar" packet sending and a description of security infringement would be provided by the analyst.

Verification should be adjusted to reproduce behaviors in a limited amount of time. For example, evidence providers should establish the exact VM environment for time-bomb, logical-bomb, and trigger-based malware to reveal their behaviors. By launching a VM on a cloud, analysts can repeat the experiments conducted by the providers of the VM who claimed that the investigating software is a piece of malware. In this way, the validity of each malware result can be established.

The CSD provides a platform for providing feedback to support comparable, verifiable, publicly available, and malware-enumerable evidence, thereby enhancing detection accuracy, resolving detection disputes, and introducing a common detection benchmark with a true malware population.

Researchers are developing new heuristic techniques and conducting experiments to tackle the challenges of malware detection, and our goal is to leverage the accumulated knowledge gradually to eventually build comprehensive malware libraries that mitigate malware uncertainties and lead toward a malware-free utopia. 

Acknowledgments

This work is supported in part by the National Science Council of Taiwan, Taiwan Information Security Center, Industrial Technology Research Institute of Taiwan, Institute for Information Industry of Taiwan, the International Collaboration for Advancing Security Technology, HTC Corporation, D-Link, Trend Micro, Promise Inc., Chungshan Institute of Science and Technology, Bureau of Investigation, and Chunghwa Telecomm. The authors thank Jeffrey Voas for his recommendations and reviews of earlier versions.

References

1. N. Perlroth, "Outmaneuvered at their Own Game, Antivirus Makers Struggle to Adapt," *The New York Times*, 31 Dec. 2012; www.nytimes.com/2013/01/01/

technology/antivirus-makers-work-on-software-to-catch-malware-more-effectively.html?_r=1&

2. L.M. Adleman, "An Abstract Theory of Computer Viruses," *Advances in Cryptology—Crypto '88*, LNCS 403, 1988, pp. 354–374.
3. G. Stoneburner, C. Hayden, and A. Feringa, "NIST Special Publication 800-27 Rev. A: Engineering Principles for Information Technology Security (A Baseline for Achieving Security)," Nat'l Inst. Standards and Technology, 2004; <http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf>.
4. C. Rossow et al., "Prudent Practices for Designing Malware Experiments: Status Quo and Outlook," *IEEE Symp. Security and Privacy*, 2012, pp. 65–79.
5. R. King, "McAfee Sees 'Malware Explosion' Across Desktop, Mobile Platforms," *ZDNet*, 22 May 2012; www.zdnet.com/blog/btl/mcafee-sees-malware-explosion-across-desktop-mobile-platforms/77531.
6. P. Baecher et al., "The Nepenthes Platform: An Efficient Approach to Collect Malware," *Recent Advances in Intrusion Detection*, 2006, pp. 165–184.
7. E. Buchanan et al., "When Good Instructions Go Bad: Generalizing Return-Oriented Programming to RISC," *Proc. 15th ACM Conf. Computer and Communications Security*, 2008, pp. 27–38.
8. L. Davi et al., "Privilege Escalation Attacks on Android," *Information Security*, Springer, 2011, pp. 346–360.
9. A.P. Felt et al., "A Survey of Mobile Malware in the Wild," *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2011, pp. 3–14.
10. C. Parampalli, R. Sekar, and R. Johnson, "A Practical Mimicry Attack against Powerful System-Call Monitors," *Proc. 2008 ACM Symp. Information, Computer and Communications Security*, 2008, pp. 156–167.
11. D. Balzarotti et al., "Efficient Detection of Split Personalities in Malware," *Proc. 17th Ann. Network and Distributed System Security Symp.*, 2010.
12. M. Christodorescu et al., "Semantics-Aware Malware Detection," *IEEE Symp. Security and Privacy*, 2005, pp. 32–46.
13. X. Jiang, X. Wang, and D. Xu, "Stealthy Malware Detection through VMM-based Out-of-the-Box Semantic View Reconstruction," *Proc. 14th ACM Conf. Computer and Communications Security*, 2007, pp. 128–138.
14. H. Yin et al., "Panorama: Capturing System-Wide Information Flow for Malware Detection and Analysis," *Proc. 14th ACM Conf. Computer and Communications Security*, 2007, pp. 116–127.

15. C. Kolbitsch, "Effective and Efficient Malware Detection at the End Host," *Proc. USENIX Security Symp.*, 2009, pp. 351–366.
16. J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software," *The Annual Network and Distributed System Security Symp.*, 2005.

Michael Cheng Yi Cho is a PhD student in the Department of Computer Science at National Chiao Tung University, Taiwan. His research interests include system security, honeypot technology, and intrusion detection. Contact him at michcho@dsns.cs.nctu.edu.tw.

Chia-Wei Hsu is a PhD student in the Department of Computer Science at National Chiao Tung University, Taiwan. His research interests include mobile security, system security, and virtual machine technology. Contact him at hsucw@cs.nctu.edu.tw.

Chi-Wei Wang is a PhD student in the Department of Computer Science at National Chiao Tung University, Taiwan. His research interests include network security, system security, and operating systems. Contact him at cwwangabc@gmail.com.

Shiuhpyng Winston Shieh is a distinguished professor and the past Chair of the Department of Computer Science, National Chiao Tung University (NCTU), and the Director of Taiwan Information Security Center at NCTU. His research interests include reliability and security hybrid mechanisms, network and system security, and malware behavior analysis. He is actively involved in IEEE and has served as the Reliability Society (RS) VP Tech, and Chair of RS Taipei/Tainan Chapter. Shieh received his PhD in electrical and computer engineering from the University of Maryland, College Park. He (along with Virgil Gligor of CMU) invented the first US patent in the intrusion detection field. He is an IEEE Fellow and ACM Distinguished Scientist. Contact him at ssp@cs.nctu.edu.tw.



Selected CS articles and columns are available for free at <http://ComputingNow.computer.org>.