# The Malware Author Testing Challenge

Tarun Moni
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
ashley.moni1@gmail.com

Sameer Salahudeen
Zighra, Inc.
Ottawa, Canada
sameer@zighra.com

Anil Somayaji
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
soma@scs.carleton.ca

***Abstract*—Attackers regularly evaluate anti-malware software to see whether or not their malware will be detected. This attacker-driven anti-malware testing is something defenders would ideally want to limit. Given that anti-malware products must be widely distributed to be commercially viable, it is not feasible to prevent attackers from running them. Here we examine whether it may be possible to instead limit the effectiveness of attacker tests.**

**Specifically, we present a game-theoretic model of anti-malware testing where detection timeliness and coverage are parameters that can be adjusted by anti-malware providers. The less coverage and the slower the response, the harder it is for attackers to determine whether their malware will be detected—and the less protection the software provides to hosts running the anti-malware software.**

**While our results are preliminary, they suggest that it is clearly non-optimal for anti-malware vendors to simply maximize coverage and detection time. As we explain, this result has significant implications for product design and (non-malicious) anti-malware testing methodologies.**

## I. Introduction

Anti-malware testing is normally framed as a problem for defenders: individuals and organizations need to choose software to defend their computers and anti-malware testing provides an empirical basis for that choice. In designing and evaluating anti-malware software, though, we must be conscious of another group that has a deep interest in anti-malware testing: attackers.

Attackers want to develop malware that can circumvent existing defenses. To achieve this goal, they have to test their software against those very defenses to see if they will be detected. It might seem that attackers would have an incentive to limit their testing given that most systems today upload novel samples to vendors for later analysis. There are a number of factors, though, that mitigate this impact. Attackers in general can easily generate malware variants, vendors have limited resources for analyzing novel malware, and many protection systems can be run successfully without any connection to the Internet. Thus in practice attackers face few barriers to testing their code against anti-malware software, even without VirusTotal (www.virustotal.com) or CloudAV [1].

When an attacker runs their code on a defended system, normally they will know whether it is going to be detected or not in a matter of seconds. If their malware is not detected, there is now a window of opportunity for attacking systems running the same defenses. This window may be measured in minutes, days, or even years; what they know, though, is that the opportunity to compromise other systems exists and can be used within that timeframe with impunity.

Another way of saying the above is that for attackers, the anti-malware testing problem is almost trivial and because of this attackers routinely bypass defenses against malware. The question we ask here is this: can we make the testing problem harder for attackers? We do not see a way to limit the ability of attackers to run anti-malware software for their own testing purposes, either directly or indirectly. We do see an opportunity, however, for making testing harder for attackers if anti-malware software doesn't classify software so quickly or accurately.

For example, consider an anti-malware application that, upon detecting a malicious piece of software, only reports this result once a month. To test against such a system attackers would have wait a month to see whether their programs were detected, greatly slowing down their ability to verify that those programs would be stealthy. We must assume, though, that targeted systems are running the same anti-malware software that attackers test. If so, then detection delay would allow attackers to do what they want on a target system for a whole month *even if it were detected*. Thus we can see a trade-off between making the attacker testing problem harder and reducing the effectiveness of the defense on targeted hosts.

In this paper we start to formalize this trade-off between anti-malware reporting timeliness and attacker testing using game theory. Our model is simple, ignoring many factors such as malware replication, difference in scale between the size of different populations (e.g., the vastly larger set of defenders versus attackers), mixed attacker deployment strategies, and scaling host subversion costs dependent upon reporting strategies. However, it does capture the basic intuition of the trade-off as described above. Further, its analysis suggests that naively reporting malware detections completely accurately and as quickly as possible is a suboptimal strategy. Our work thus suggests that "reducing" anti-malware performance could in fact make the development of zero-day attacks much more difficult, and that such reductions might be possible without placing defended systems at undue risk. While these results should be considered to be preliminary, they are suggestive enough that we believe further study into the attacker anti-malware testing problem is warranted, both in theory and in practice.

The rest of the paper proceeds as follows. In Section II we define the problem we are attempting to model. Section III presents our simple game-theoretic model of testing interactions between attackers and anti-malware providers. We analyze potential implications of our model in Section IV and discuss related work in Section V. Section VI concludes.

## II. The Problem

Our first goal is to try to model the overarching problem formally. There is a large system of agents (attackers, normal users, anti-malware software providers), a space of malware to be designed, and a space of defensive software solutions (with associated reporting strategies).

All of the agents are each trying to optimize their own goals while stymying or assisting each other based on how convergent those goals are. Their actions have probabilistic outcomes. From here it seems reasonable to model the problem using game theory, the problem being a zero-sum game between the attackers on one side versus the average users and anti-malware providers on the other.

The most general problem is then a large game where three populations of players play: attackers attempt to develop malware to infect normal users (hereafter referred to as "defenders"), anti-malware defense providers develop and disseminate software to defenders to defeat the malware, and defenders use the software given to them by their defense providers to the best of their informed ability to protect themselves.

An important constraint, perhaps the *most important* constraint, is that defense providers cannot differentiate between attackers and defenders. Attackers will also be given defensive software, and will exploit it to further their own agenda.

There is merit in modelling the network consequences of subverting defender systems. The utility of a set of systems being infected can be greater or lower than the sum of utilities for infecting each individual system (botnets, self-reproducing malware, etc.). Still, a reasonable simplifying assumption is to ignore this and focus on subsets of the population that still emphasize the problem at hand.

Consequently we will reduce our consideration to the smallest subset of the player population that still characterizes the problem: three agents and two computer systems. An attacker and a defender fight for control over the defender's system, both using tools given to them by a single anti-malware defense provider.

To help define the game they play and the strategies they can formulate, we will model the space of all malware, $X$. $X$ is the set of all possible exploits that a malware can use to successfully attack a host system. There is a surjection from the space of all malware onto the space of all exploits: Every piece of malware will be mapped to a single element of $X$ (each malware will only use a single exploit), while any exploit which has no possible corresponding implementations will be ignored.

**The defense provider** will have access to a subset of $X$ called $K$; these represent known exploits. For every element within $K$ the defense provider will be able to build software which can detect and eliminate all associated malware. The defense provider can also construct software that is only capable of detecting subsets of $K$; these subsets will be referred to as *coverage*. After designing each software instance, they will be randomly assigned to the attacker and defender (the software need not be identical, the other players are simply indistinguishable). Neither attacker nor defender know the full extent of $K$; they only know what is covered by the defense software they are given.

**The attacker** can generate malware via development; we make the simplifying assumption that the attacker has no control over which element she 'selects' from $X$. Every time she develops a new malware program, it uses a random exploit from $X$ (using a uniform distribution) and can be traced using that exploit's signature (if a defender system has coverage over the exploit, the malware can be found).

**The defender** herself simply uses the tools given to her by her defense provider. The defense provider cannot distinguish between her and the attacker for the purposes of offering anti-malware coverage. The defender generally has a simple optimal strategy of using the anti-malware software she has been provided as frequently and as totally as possible with no considerations of restraint.

The game is zero-sum and continuous (on various variables ranging from coverage schemes chosen to mixed attacker strategies for discarding malware and developing new versions). The defense provider first constructs a coverage model, which is a scheme for distributing subsets (or perhaps the entirety) of $K$ to both the attacker and defender indiscriminately. The attacker can then use the coverage offered to her to help test and dismiss malware she designs: gathering information on the probability that she has a genuine zero-day exploit (element outside of $K$), or assessing the chance that her program can bypass the limited coverage offered to her target. After the attacker settles on a final program, it's deployed and utility is gathered by all players based on pre-defined discrete payoff coefficients (malware captured by defender, zero-day developed, etc.).

## III. The Model

We'll make a few assumptions, some taken directly from the introduction, and design a simple instance of the more general game formulation above:

1) The cost for the attacker to discard any given malware and build a new one is low.
2) The defense provider uses a 'random noise' coverage model, where they offer a fixed percentage of their signature database $K$ to all agents in the population (defenders and attackers). Since the signatures offered to everyone are random, the coverage set for each system is mutually independent.
3) Coverage rotates (or more accurately, re-randomizes to a new subset distribution of $K$) on a slow enough timescale that the attacker cannot afford to wait and test a given malware against multiple coverage iterations.

Each assumption has a specific purpose. The random noise model is an extremely simple realization of 'partial coverage' that can be easily scaled into modern total coverage systems. This way we can directly compare the value of partial coverage to our modern total coverage strategy. It also helps us describe

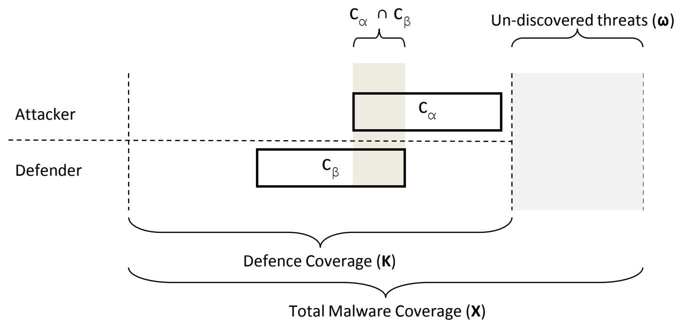

Fig. 1. The game space: $c_\alpha$ and $c_\beta$ represent the local coverage offered to the two anonymous players by the defense provider, while $\omega$ is the space of zero day exploits (the set difference $X \backslash K$).

the game using simple probability theory. The low-cost development and slow-rotation assumptions help constrain the attacker to a simple optimal strategy: Always dismiss a given program if the attacker's local coverage catches the program, otherwise deploy.

The defense provider offers both other players random subsets of $K$ with a fixed measure $c$. Their coverage (labelled $c_\alpha$ and $c_\beta$) overlaps in parts, stand alone in others, and neither cover a final remaining segment of $K$. No coverage set can intrude upon the set of zero day exploits outside $K$ (labelled $\omega$) as it is beyond the knowledge of the defense provider; only the attacker can interact with this region. The relationship between these variables is illustrated in Figure 1.

We shall normalize the measure of the set $K$ to 1. This lets us treat the public coverage percentage $c$ and the measure of known exploits covered interchangeably. The measure for the shared coverage $c_\alpha \cap c_\beta$ of both attacker and defender is $c^2$ since they are independent.

There are four possible results to this game. Each will be ascribed its own utility coefficient:

1) $u_z$ is the utility of the attacker stumbling across a zero day exploit. The exploit they've randomly selected falls within $\omega$ and cannot be caught by the provider's complete signature database.
2) $u_r$ is the utility of the attacker's malware being caught by their local system before being deployed in the wild.
3) $u_c$ is the utility of the attacker deploying malware that is then caught by the defender's system.
4) $u_s$ is the utility of the attacker deploying malware that is *not* caught by the defender's system, but ultimately still falls within $K$ and will be eventually caught by coverage rotation.

From the perspective of the defense provider $u_z$ and $u_s$ will be negative, $u_r$ will be negligible and $u_c$ will be positive.

The aggregate utility function that the defense provider is trying to optimize can be found by taking the sum of products of each utility coefficient and its respective probability.

$$U(c) = u_z\omega + u_r c + u_c(c - c^2) + u_s(c^2 - 2c + 1)$$

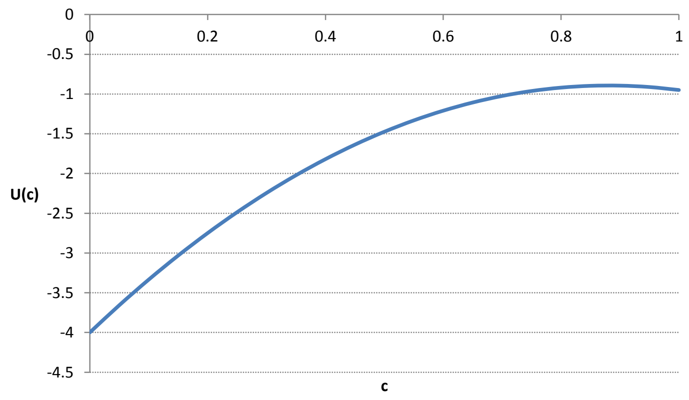

Fig. 2. $U(c)$ maximum at $c = \frac{141}{160} \approx 88\%$.

Using the following sample values for each utility: $\{\omega = 0.1, u_z = -10, u_r = 0.05, u_c = 1, u_s = -3\}$ we can generate the graph Figure 2.

The maxima is not at $c = 1$; it's a bit lower. A general solution for this maxima can be derived from the aggregate utility function and is as follows:

$$c_\uparrow = \frac{u_c + u_r - 2u_s}{2(u_c - u_s)}$$

where $c_\uparrow$ is the value of c which will generate the largest $U(c)$.

This can be restated as follows:

$$c_\uparrow = 1 - \underbrace{\left(\frac{u_c - u_r}{2(u_c - u_s)}\right)}_{\text{shroud factor}}$$

The fractional component shall be referred to as the *shroud factor* henceforth.

As long as we make the following assumptions:

1) $u_c > 0$ (It's worth catching their malware)
2) $u_s < 0$ (It's bad to knowingly let a host system be compromised)
3) $u_c > u_r$ (Catching a given malware is better than forcing the attacker to develop a new one)

…we can prove that the shroud factor will always be positive. Since we've already assumed that $u_r$ is negligible, this isn't a very difficult conclusion to draw at all.

As long as the shroud factor is positive, $c_\uparrow < 1$ and the defense provider's optimal strategy is to hide a random subset of $K$ from all clients.

## IV. Implications

The shroud factor captures how much of provider's protection abilities—its database of signatures and heuristics and, potentially, the capabilities of its human analysts—is to be hidden from all users in order to mitigate the ability of attackers to develop new malware that bypass current defenses. If attackers wish to be stealthy, the ability to quickly assess whether their attack will be detected by current anti-malware software is extremely valuable, as it ensures that their stealth failures mostly occur on hosts that they control.

This notion of not revealing all of a defender's capabilities to an attacker is not new. It has long been know in the intelligence community that capabilities should often be hidden even when other considerations (such as harm to individuals or standards of due process) would normally require those capabilities to be revealed. While watching an attacker compromise an organization and exfiltrate data might be a useful strategy for a government entity engaged in a long-term struggle for information dominance, it is not obvious if the same considerations make sense for anti-malware providers. What the shroud factor suggests is that such concealment does, in fact, make sense.

But what form should that concealment take? Should it be a simple strategy such as randomly reducing the number of detectors each host uses, or should it be a more targeted strategy where particularly novel malware is allowed to spread for some time while it is observed by a team of experts? Our model is not precise enough to differentiate between different concealment strategies. We suggest that automated approaches hold the most promise due to their inherent scalability on the defender side.

One such simple automated strategy would be to have the defense system trigger a full coverage scan at a random frequency. The frequency could be randomly determined based on the period elapsed from the last full coverage scan and the number of partial scans being carried out during that period. This full coverage scan would be totally out of control and awareness of the user, assuring that the adversary will not be able to only test their malware during a full scan. Regular users should be made aware of this pattern of scans, however, as that way attackers will know that straightforward detection tests cannot be trusted. Indeed, a key purpose of such a strategy is to introduce uncertainty into the planning of attackers, thus forcing them to move much more slowly in their development than they would otherwise. This benefit in itself may be the key motivation for anti-malware providers to pursue detection concealment strategies.

One unfortunate side-effect is that any such concealment efforts will also have a negative effect on anyone evaluating anti-malware software for legitimate ends. Results from any lab test can no longer be trusted as part of the results may be concealed. Proper testing would now require repeated evaluation over an extended period of time. While such testing is possible in a lab context, widespread shrouding of provider capabilities would likely necessitate the use of more ecologically valid testing such as the clinical trials methodology [2], [3]. The cost of such a shift would be significant; if this shift is part of an industry-wide effort to make systems less vulnerable to zero-day threats, though, these costs may be justified.

## V. Related Work

While our focus on attacker anti-malware testing is novel, many researchers have the dynamics of attackers and defenders in a computer security context. Much of this work has a biological flavor of some kind. The very term "computer virus" [4] evokes the competition between living things, and as such models of attackers and defenders in computer systems have often looked to biology for inspiration. Epidemiological modelling of the spread of malware done for famous incidents such as Code Red [5] and for more generally understanding the spreading patterns of email viruses [6]. Attacker-defender modeling of malware has also been key to immunological approaches to computer defenses such as Kephart's immunological kill signals [7].

Dynamically changing coverage has been suggested as part of general design principles for computer immune systems [8] and was implemented as part of the LISYS network intrusion detection system [9], [10]. It is also a general feature of diversity/randomization approaches to low-level process behavior such as ASLR [11] and instruction set randomization [12], [13]. We should note, though, that "rolling coverage" in this work is normally seen as an inevitable consequence of other design decisions (such as randomization or distributed detection) rather than as an effort to reduce the ability of attackers to find gaps in protective mechanisms.

While game theory has a well-established place in computer security [14], particularly in the context of economic models of computer security [15] and information warfare [16], to our knowledge this work is the first attempt to apply it to the problem of attacker anti-malware testing and, more generally, the issue of whether defenders should conceal their capabilities in a civilian, commercial anti-malware provider context.

## VI. Conclusion

The wide availability and uniform behavior of anti-malware software allows attackers to benchmark their attacks against the current protections easily and efficiently, thereby facilitating the production of stealthy zero-day attacks. In this paper we present a simple game-theoretic model of attacker/anti-malware provider interactions that suggests that providers can reduce the ability of attackers to easily create stealthy attacks through "shrouding" their coverage. By increasing the uncertainty as to whether their malware will be detected, attackers must devote significantly greater efforts to testing their malware. This additional attacker effort potentially helps balance the playing field between attackers and defenders.

Although we proposed some ideas, we do not yet have effective strategies that minimize the exposure of legitimate hosts while maximizing the attacker effort required to develop new undetectable malware. We believe that both theoretical and practical work on such strategies are promising areas for future work.

## Acknowledgements

The first and third authors would like to acknowledge the support of Canada's Natural Sciences and Engineering Council (NSERC) through their Discovery grants program.

## References

[1] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud." in *17th USENIX Security Symposium*, 2008.

[2] A. Somayaji, Y. Li, H. Inoue, J. M. Fernandez, and R. Ford, "Evaluating security products with clinical trials." in *CSET*, 2009.

[3] F. Lalonde Levesque, J. Nsiempba, J. M. Fernandez, S. Chiasson, and A. Somayaji, "A clinical study of risk factors related to malware infections," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 97–108.

[4] F. Cohen, "Computer viruses," Ph.D. dissertation, University of Southern California, 1985.

[5] D. Moore, C. Shannon, and k. claffy, "Code-red: A case study on the spread and victims of an internet worm," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. New York, NY, USA: ACM, 2002, pp. 273–284. [Online]. Available: http://doi.acm.org/10.1145/637201.637244

[6] M. E. J. Newman, S. Forrest, and J. Balthrop, "Email networks and the spread of computer viruses," *Phys. Rev. E*, vol. 66, p. 035101, Sep 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.66.035101

[7] J. O. Kephart *et al.*, "A biologically inspired immune system for computers," in *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*, 1994, pp. 130–139.

[8] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proceedings of the 1997 workshop on New security paradigms*. ACM, 1998, pp. 75–82.

[9] S. A. Hofmeyr and S. Forrest, "An immunological model of distributed detection and its application to computer security," Ph.D. dissertation, Citeseer, 1999.

[10] J. Balthrop, S. Forrest, and M. R. Glickman, "Revisiting lisys: Parameters and normal behavior," in *Computational Intelligence, Proceedings of the World on Congress on*, vol. 2. IEEE, 2002, pp. 1045–1050.

[11] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 298–307.

[12] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, and D. D. Zovi, "Randomized instruction set emulation to disrupt binary code injection attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 281–289.

[13] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 272–280.

[14] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE, 2010, pp. 1–10.

[15] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 4, pp. 438–457, 2002.

[16] D. A. Burke, "Towards a game theory model of information warfare," DTIC Document, Tech. Rep., 1999.