Misleading Modern Malware

Daniel Bilar

Wellesley College
Department of Computer Science
Wellesley MA 02481, USA
dbilar@wellesley.edu

Abstract. For practical purposes, information-gain adversarial malware may soon become undetectable using current function-based signature-matching AV techniques. We propose and sketch defenses that adopt an interactive approach, based on controlling information-centric Kullback-Leibler distances.

1 Introduction

We shall discuss the question how to tackle the detection of highly evolved, modern malware. In light of results which suggest the practical and perhaps theoretical limitations of traditional 'white-box' AV approaches, we propose moving beyond predominantly byte sequence-matching white-box AV premised on classic Turing Machine, function-based assumptions towards iterative games and black-box process modeling, as expressed by interactive computing models.

The rest of this paper is structures as follows: Sec. 2 gives a short overview of modern malware characteristics. Sec. 3 the assesses current AV's capability to handle said malware. Sec. 4 describes additional environmental factors and structures which work in modern malware's favour. Sec. 5 describes passive target techniques and active defense frameworks based on controlling the malware's information gain. Sec. 6 briefly discusses the direction and implication of this approach.

2 Malware

Polymorphic malware contain decryption routines which decrypt encrypted constant parts of the malware body. The malware can mutate its decryptors in subsequent generations, thereby complicating signature-based detection approaches by limiting the 'constant base' to the decryption routine and the occasional odd invariant in the body.

Metamorphic malware generally do not use encryption, but are able to mutate their body in subsequent generation using various morphing techniques, such as junk insertion, semantic NOPs, equivalent instruction substitution, code transposition, subroutine permutation, incremental stack construction, and register reassignments [1–3]. For a recent formalization of these code mutation techniques, the technical reader is referred to [4].

K-ary malware, of which at this time only laboratory or very trivial examples are known to exist, partition their functionality spatio-temporally into k distinct parts, with each part containing merely an innocuous subset of total instructions. In serial or parallel combination, they subsequently become active [5].

Information-gain adversarial malware It is our contention that modern malware systematically thwarts information-theoretical entropy reduction. The implementation characteristics of this design philosophy include massive generation of functionally equivalent phenotypes, sophisticated UEP (unknown entry point), anti-emulation, anti-disassembly, code integration, substitution, decoy and subroutine permutation techniques, within a time-dependent, multi-stage structure. We will term this abstract type information-gain adversarial malware.

We will emphasize its techniques' pronounced information-theoretical dimension: The systematic adversarial design of this breed of malware strives to reduce the relative information gain of the defender: Given constant static analysis time, there is increasing uncertainty about the location, control flow handoff and activation triggers, and even existence (through code integration diffusion) of malware. Similarly, given constant dynamic analysis time, through dummy loops, k-ary design, anti-VM and anti-emulation countermeasures, modern malware systematically lessens the hoped-for information gain of these defensive emulation techniques. We stress that the reduction needn't be complete, or even unmitigatable under more relaxed resource constraints. Our concern are the practical, day-to-day, near-real-time demands on these defensive techniques.

3 Anti-Virus

Estimated malware-attributed economic damages have been hovering around $14b/year for the last seven years [6]. During the same time span, the host base (end systems with IP addresses) grew from 93m to roughly 490m [7]. In conjunction with stagnating damages, these numbers could plausibly be interpreted as a success story for signature-based anti-viral (AV) software, nowadays routinely deployed on end systems.

Commercial antivirus (AV) products rely mostly on signature matching; the bulk of which constitutes strict byte sequence pattern matching. AV vendors have developed some static and dynamic analysis enhancements; the former based on DFAs (wildcards, regular expressions), the later on heuristics such as emulation and runtime analysis. For detection and mitigation of modern malware, however, these AV techniques will soon hit their practical and maybe theoretical limits.

3.1 Empirical Detection

Recent empirical AV detection rates for recent modern malware do not look reassuring. In February 2007, for instance, seventeen state-of-the-art, updated AV scanners were checked against twelve well-known, previously submitted, highly poly- and metamorphic malware samples. The miss rate was 100% to 0%, with an average detection miss rate of roughly 38% [8]. The capabilities and general detection rates of variants produced by over five to ten year old, freely available hobbyist engines such as the Mistfall (W32/Zmist), MetaPHOR (W32/Simile), and more recently, the code-building MSIL metamorphic engine (MSIL/Gastropod), as well as 150+ other kits and mutators should give some pause [1, 9].

Polymorphic malware identification may be bolstered by indications that certain structural homogeneities viz non-malicious software are sufficiently disturbed by encryption to be detectable by statistical entropic differential analysis [10, 11]. Ominously, recent results also indicate that current AV seem unable to detect within a reasonable time frame proof-of-concept k-ary malware, or even disinfect the system completely after detection occurred [5].

Some industry experts continue to maintain that heuristics enhancements such as emulation and runtime analysis will keep up with modern challenges [12]. Recent research, however, showed the general limitations of highly sophisticated classifiers [13] against data which exhibit information-gain adversarial traits such as worm traffic and email spam [14].

3.2 Theoretical Detection

It should be noted that current AV approaches could try to leverage their power by normalizing [15, 2] and second-order transformation detection techniques [16], and as such potentially address some present-day metamorphic obfuscation [17]. Though theoretical detection concerns have been recognized early on [18] and emphasized recently [5], they seem under-appreciated (maybe forgotten) in practitioner's circles. It is quite likely that undecidability [19] and computational complexity issues [20] will soon be at odds with mundane practical constraints of tractable detection, as has recently been reported [5].

From an AV modeling point of view, we also may find the Turing machine model (equating computation as such with mathematical closed-box transformations of input to output [21]) insufficiently expressive to capture this type of information-gain adversarial malware[1]. If this radical conjecture is true, Turing machine models premised on the (strong) Church-Turing thesis (computation-as-functions) may have to give way to more expressive models, such as those

[1] Interestingly, the necessity of theoretical model evolution was foreshadowed by Turing in his 1936 paper [22] with his choice 'c-machine', as opposed to the standard automatic 'a-machine'

based on interactions. We shall pick this up in more detail in Sec. 5.3, as we now turn our attention to the target of modern malware.

4 Digital biotopes, habitats and predators

The Internet consists of a substrate of connected physical networks (education, commercial, governmental, country, etc). On a logical level these separate networks can minimally communicate via the IP network layer protocol. Recent years also witnessed the rise of mobile, wireless IP via bridged cellular networks. This logical IP substrate is overlayed (mostly) transparently with other logical networks that serve a variety of purposes (such as DNS, P2P, IRC/IM, fileshares, email, surface WWW, dark WWW with database backend, router, online gaming, etc) which communicate mostly via higher-level protocols.

These logical networks can roughly be viewed as the digital equivalent of natural biotopes. In keeping with the ecological simile, they serve as 'habitats' for a variety of specialized software. Habitats give rise to predators and parasites; malware can be seen as the rough digital equivalent. Two distinctive features of the digital biotopes and its denizen give predatory types of programs an advantage over similar predator-prey setups in the natural world: *Monoculture* as a result of engineered design processes and *mutation time scales.*

Our global network infrastructure was consciously designed by humans. It did not evolve 'blindly' in any fundamental sense like natural systems. The primary objective - at least viz the topology - was to ensure communication survivability, its subsequent development was geared towards the seamless exchange of data between trusted sources and, to a lesser extent, generativity [23]. Early systemic host diversity requirements would have likely impeded the explosive growth of the global network infrastructure, as would have network-wide immune system with the added extra layers of complexity and cost.

These qualitatively different selection criteria (together with explicitly designed objectives) mark a crucial distinction between semi-evolved/engineered electronic network system and the ecological system that informed our analogy. It has been argued from first principles, for instance, that the Internet router topology is the product of a generative HOT (Highly Optimized Tolerance) mechanism [24].

HOT mechanisms are processes that induce highly structured, complex systems through processes that seek to optimally allocate resources to limit event losses in an probabilistic environment [25]. The induced systems tend to be robust towards anticipated perturbations but fragile towards the unknown/unexpected; an important characteristic that benefits both predator and prey to a discussion of which we shall return later.

Our global engineered networks are homogenous in a way that natural selection would have selected against in ecological networks. In natural biotopes, there is enough random fluctuation of the constitutive elements (people, organisms, geography and the environment) to dampen the effects of most failures. For instance, the 1918 influenza outbreak after WW1, the often forgotten but the most devastating pandemic in human history (50m-100m deaths [26]) did not wipe out humanity. Instead, the airborne, highly infectious disease retreated after ravaging most of the globe. Variations in individuals, natural population movement, physical isolation prevent most catastrophic failures from cascading. When humans impose homogeneity, catastrophe can ensue: Corn and cotton monoculture in US agriculture is blamed for devastating losses by disease and predators in the 20^{th} century [27, 28].

In networks, we got a preview with the 'Code Red' worm suite (July-August 2001), one version of which managed to infect over 350,000 Microsoft IIS servers in a 14 hour period [29]. The potential security implications of software monoculture was studied relatively late in the network life cycle. Addressed by Forrest in 1997 [30], an unheeded paradigm shift in favour of heterogenous networks was proposed in 2001 [31] and picked up by Geer et al [32] again in 2003.

These assertions are not specious: See the market leader's share for the host (PC) biotope (2006: 65%-85% [28]); web biotope (2006: 60% run Apache's `httpd` [33]); DNS biotope (2004: 75%-95% run `bind`, including the 13 root servers); mail biotopes (2001: 42% run `sendmail`); database biotope (2005:44% run `mysqld`), and the list goes on [34].

Compounding this homogeneity-induced susceptibility is the relative quicker evolvability (and soon autonomous adaptability) of digital predators.A back-of-the-envelope calculation for cultured bacteria illustrates this point: So-called 'generation times' for most cultured bacteria - growth rates during the exponential phase - vary from about 15 minutes to 1 hour [35]. A modern measure for phenotypical variability is the 'haldane' which indicates phenotypic standard deviation per generation. On the time scale of one generation, the rates are 0.1-1 haldanes [36], this corresponds roughly to 4%-30% change per generation (rates changes over different time scales, a fascinating subject in its own right). However, because biological evolution is blind and random (subject to selection pressures), not all mutations will be viable, nor improved the bacteria.

This is not the case for malware engineering which is both purposeful (functionality is preserved) and algorithmic (viability is preserved). Assuming conservatively that variants can be generated at a rate of maybe 10-1000 mutations/minute, depending on the complexity of the binary, this means that digital predators can evolve viable offspring at a rate of at least two to five orders of magnitude faster than their ecological counterparts (theoretically ad infinitum [37]).

Add to the witches' brew of information-gain adversarial techniques the ability to jump across habitats (for multi-habitat malware, see the proof-of-concept PC-to-PDA 'Crossover'[38]) and certain certain topological features of networks [39], and it is feasible that before long, the Internet will have undetectable, permanently roving diseases. The first generations may be containable through concerted, coordinated efforts, but subsequent ones may be undetectable via present means, and for all intents and purposes, ineradicable. Note that the email-born Bagle/Beagle worm (which first appeared in 2004) is already a plausible contender: 30'000 distinct variants server-side supplied in 2007 alone, an average of 625 new variants a day [40, 41]. It seems prudent to shift and adapt some defense resources to specifically counter this type of information-gain adversarial malware.

5 Practical directions towards information-centric defenses

Information-adversarial design strives to reduce the relative information gain of the defender's strategies. Defenders have to enrich their byte pattern matching techniques by adopting a similar Bayesian, information-centric viewpoint, controlling both entropy of the prior distribution, as well as the information gain in the a posteriori distribution. Some defenses will *actively* engage in an iterative 2-player (possibly n-player), imperfect, non-zero-sum game in order to control the relative information gain of the malware's reconnaissance and influencing its decision algorithms.

This active approach bears some similarities to the concept of subverting an enemy's OODA (Observe, Orient, Decide, and Act) loop, an information warfare strategy pioneered by military fighter pilot Col. John Boyd which seeks to pro-actively influence and change enemy behavior [42]. The interplay between observation, hypothesis generation, feedback and analysis is illustrated in Fig. 1 from [43] .

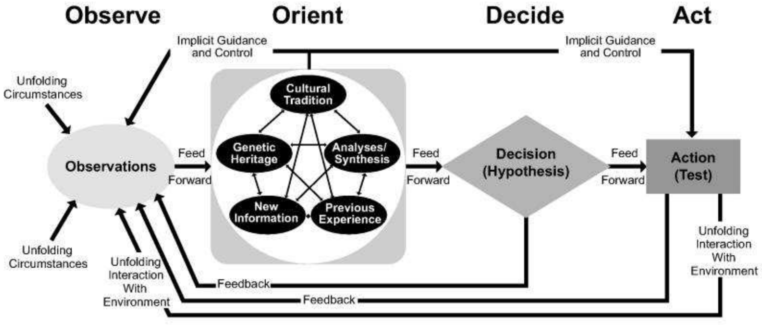


Fig. 1. OODA Loop

In our context, defenses may also be expressed *passively* and target the prior distributions. This passive tack seeks to lessen exploit success by introducing heterogeneity in the digital biotopes. Hence, informally, through decoys and irregularities, interactions and observations, defenders probabilistically implement strategies that actively and passively lead the malware astray.

5.1 Controlling Information Gain

On an abstract level, we propose manipulating the Kullback-Leibler distance [44] for observational decision hypotheses H_1 over H_0, in favour of the defense. We stress *control*, not minimize. The goal is not primarily to confuse the malware by keeping entropy as high as possible (although under some situation minimization is a strategy, see illustration); the goal is to enter the opponent's decision loop (control flow) and similar to chess, lead the opponent down the path that is most amenable to victory. We have

$$D_{KL}(p(x|H_1)||p(x|H_0)) = \sum_i p(x_i|H_1)\log\frac{p(x_i|H_1)}{p(x_i|H_0)} \tag{1}$$

where

$$\begin{aligned} x &\equiv \text{observation made by malware} && (2)\\ p(x|H_1) &\equiv \text{conditional probability of observation x given } H_1 && (3)\\ p(x|H_0) &\equiv \text{conditional probability of observation x given } H_0 && (4) \end{aligned}$$

Illustration A toy illustration is in order. Say the malware is scanning a subnet N. Let $x_i = 1$ if a ICMP response is received from host i, $x_i = 0$ otherwise. Let H_1 be the worm's hypothesis (used internally for control flow decisions) that host i is a target, H_0 that host i is not a target. Assume the malware receives a response $x_j = 1$ for a given host j. We have

$$\begin{aligned} D_{KL}(p(x_j|H_1)||p(x_j|H_0)) &= p(x_j|H_1)\log\frac{p(x_j|H_1)}{p(x_j|H_0)}\\ &= p(x_j=1|H_1)\log\frac{p(x_j=1|H_1)}{p(x_j=1|H_0)}\\ &= p(x_j=1|H_1)\left(\log p(x_j=1|H_1) - \log p(x_j=1|H_0)\right) \end{aligned}$$

where

$$\begin{aligned} p(x_j=1|H_1) &\equiv \text{probability of getting a response, given } j \text{ is a target}\\ p(x_j=1|H_0) &\equiv \text{probability of getting a response, given } j \text{ is not a target} \end{aligned}$$

Now consider three sample defense strategies, S (no defense), S_{HNP} (honeypot), S_{FLT} (filter/block ICMP response) and the effects on the D_{KL} distance used by the malware for further action in Table 1. Here, we controlled the information

Strategy	$p(x_j = 1\|H_1)$	$p(x_j = 1\|H_0)$	D_{KL}	Interpretation
S	1	0	$1(1-0) = 1$	H_1 (viable host) is favoured over H_0
S_{HNP}	1	1	$1(1-1) = 0$	No information gain to favour H_1 over H_0
S_{FLT}	0	0	$0(0-0) = 0$	No information gain to favour H_1 over H_0

Table 1. Malware information gain under three defense strategies

gain of the a posteriori distribution. Both S_{HNP} and S_{FLT} defense strategies make it impossible for the malware to favour one hypothesis over the other, on empirical grounds. We have hence influenced the malware's control flow, and as such, entered and controlled its OODA loop. We shall now discuss passive and active implementations adopting this information-centric point of view.

5.2 Passive Defense Techniques

Passive techniques that affect the prior belief distribution have been developed for some time now, even though their deployment is relatively recent.

Honeypots and honeynets [45] - simulated decoys that detract from 'real' networks, hosts and services - are probably the best-known examples though this was not their primary *raison d'être*. [46] implement a recent highly scalable cleverly parsimonious hybridization of low- and high-interaction honeynets that doubles as a platform for malware collection.

On the executable level, and leveraging the fact that exploits are highly sensitive to unexpected disturbances in the exploit environments (for an illuminating example of the highly complex dependencies between user/kernel processes and threading, see the Slammer conditions[47]), ad-hoc hot patching techniques such heap, stack, and format string mutations [48] have been proposed. More mature techniques include various address space layout disturbances, such as random heap, stack, library positioning at compile, link and load times [49].

Homogenizing the a priori belief distribution of defense mechanisms also falls under this rubric; for an example of detection classifier randomization with remarkable empirical results, see [50]. It is somewhat heartening that the very same generative optimization processes[25] that induce executable structures with fragility viz unexpected perturbations (see [51] for a buffer overflow vulnerability example) can be leveraged against the exploits that target them. In the long run, a framework of Matryoshka-esque VMs may have to complement these passive techniques.

5.3 Active Defense Framework

We seek to manipulating the malware's view of the world. This entails modeling its internal hypothesis structure, entering its OODA loop and hence controlling its decisions. To this end, we need an *observation framework* that can infer said internal hypothesis structure and a *control framework* that dynamically chooses strategies which control adversarial information gain for the benefit of the defender.

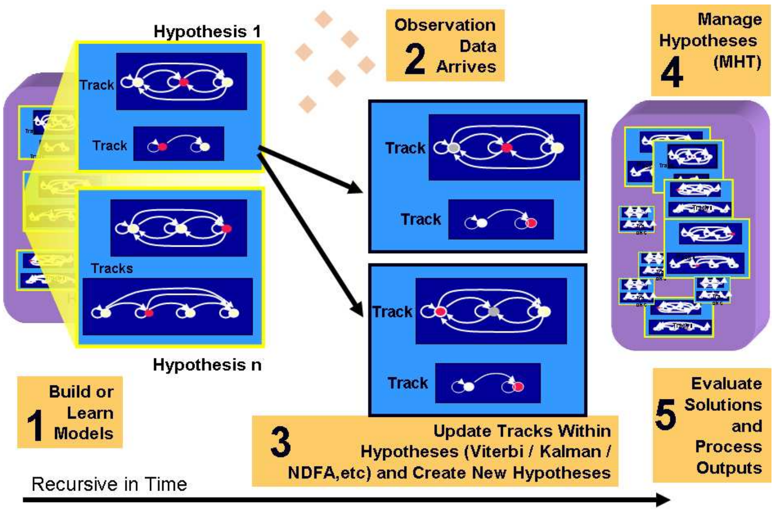


Fig. 2. PQS operation sequence

Process Query System PQSs were initially designed to solve the Discrete Source Separation Problem by setting up a DBMS framework that allows for *process description* queries against internal models, a task for which traditional DBMS are unsuitable, since the queries (e.g. SQL) are typically formulated as Boolean expressions [52]. These models can take the form of Finite State Machines, rule sets, Hidden Markov models, Hidden Petri Nets, among others.

Four sequential components are linked together in a PQS [53]: Incoming observation → multiple hypothesis generation → hypothesis evaluation by models → model selection. The overarching goal is to detect processes by leveraging the correlation between events (such as observations) and the processes's states.

Fig. 2 from [54] illustrates these components. Processes have hidden states which emit observables. The relationship between observables and states is not bijective, meaning a given observation may be emitted by more than one state. The so-called 'tracks' are associations of observations to processes. Hypotheses represent consistent tracks that explain the observables. The hypotheses in our domain correspond to the malware's internal control structure, which is inferred from its behaviour through observation. We propose that a PQS serve to dynamically ‘black-box model’ modern malware. The necessary observation events can be both passively recorded and actively enticed through iterative interactions.

Interactive Computations The notion of computability rests largely on the Church-Turing thesis. Although the Church-Turing thesis refers explicitly to the computation of *functions*, this small but important caveat is not emphasized to the extent that it should be; instead it is understood to mean that Turing machines model *all* computation [55]. Herein lies the flaw: Not only are there some functions that cannot be computed by Turing machines (see Rado’s Busy Beaver[56]), but more fundamentally, not all computable problems are function-based to begin with.

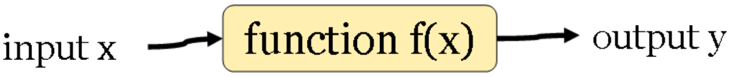


Fig. 3. Computation as a function-based, closed transformation from input to output.

Function-based or algorithmic (with loops) computation requires the input to be specified at the start of the computation; in other words it is a closed transformation from input to output (Fig. 3). Knuth’s recipe “toss lightly until the mixture is crumbly” is therefore not algorithmic because it requires feedback, an interaction with the environment, to determine when to stop.

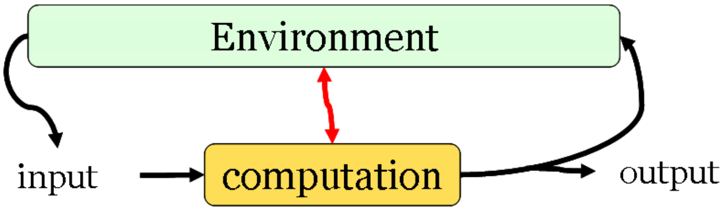


Fig. 4. Computation as a interactive, open transformation from input to output.

In contrast, interactive computational models serve as a theoretical bridge between Turing Machines (functions) and interaction (communication) with the environment. The computation is open, I/O happens during computation, not just before or after (Fig. 4). This model describes everyday computing more accurately than closed transformation, since a GUI, an OS, a Control System does not ‘compute’ in the strict closed algorithmic sense of computing.

An informal discussion of computing problem solving, a more extensive elaboration of above argument, as well as a canonical reference on Interactive Computing can be found in [21, 55, 57].

Iterative imperfect non-zero-sum games The goal of the control framework is to create the illusion of win-win (non-zero-sum) viz the malware's goals by iteratively either weakening useful/accurate and strengthening useless/misleading information gain through defensive strategies. Game theory provides a suitable interactive framework. An expansive treatment is outside of the scope of this text, see [58, 59] for a canonical text, as well as online resources at all levels and domains.

KL over time Iterative games imply that the information gain measure has a temporal dimension Δt. We refine Equ. 1

$$\frac{D_{KL}(p(x|H_i)||p(x|H_j))}{t_k}, i \neq j \tag{5}$$

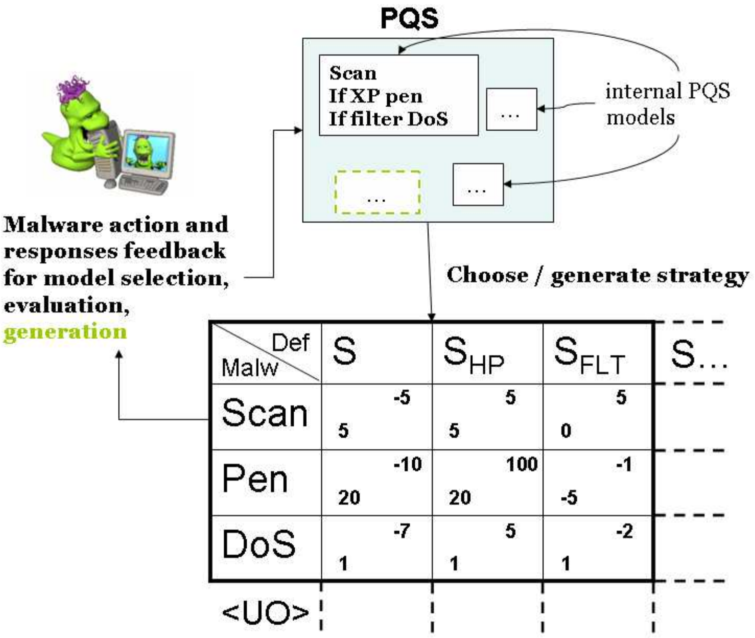
where

$$t_k \equiv \text{time differential between round } r \text{ and } r\text{-}1 \ , t_r - t_{r-1}$$

Equ. 5 can be used as an objective function in a dynamic optimization framework to choose strategies that control the adversarial information gain at each step. Tarpits[60] capture the 'temporal control' flavour of this technique. The Siren system [61] offers an example of a strategy in this context: For the purpose of thwarting mimicry attacks, Siren tries to coax the adversary into producing a known sequence of network requests. Game-theoretical approaches for IDS have been studied extensively [62, 63]. Strategy sets may amount to reverse Turing tests: The CAPTCHA thwarting of automated bot signups is probably the most familiar example [64]. Recently, NEC has developed technology that can tell humans from computers to combat VoIP phone spam[65].

5.4 Illustration

Our active defense framework is sketched in a toy example in Fig. 5. Suppose the malware's toy internal hypothesis structure and strategies are modeled by `Scan;if XP penetrate;if filtered DoS` in a PQS internal model. The defense's strategies are S (no defense), S_{HNP} (honeypot), S_{FLT} (filter/block ICMP response). The game matrix shows the payoffs of the defense's and malware's strategy combinations. The malware starts scanning and wants to get to $[Pen, S]$ penetrating a real host. The defense wants to engage sequential strategies such that the malware penetrates a fake host $[Pen, S_{HP}]$, thereby giving the illusion of a win for the malware while learning more about it. Again, the defense wants to iteratively control, not necessarily minimize the malware's $D_{KL}(p(x|H_i)||p(x|H_j))$. Strategies may not be fixed and dynamically generated as PQS models adapt to the malware responses, as denoted by $S_{...}$ (new defense strategy) and $< UO >$ (unknown observation).


PQS
Scan
If XP pen
If filter DoS
…
internal PQS models
…
…
Malware action and responses feedback for model selection, evaluation, generation
Choose / generate strategy
<UO>

Def Malw	S	S_HP	S_FLT	S…
Scan	-5 5	5 5	5 0	
Pen	-10 20	100 20	-1 -5	
DoS	-7 1	5 1	-2 1	

Fig. 5. Active Defense: Observation/Modeling through PQS, Control/Response through interactive game strategies

Thus, our proposed interactive, OODA-loop subverting strategic 'judo' against modern malware marks a philosophical shift. From from predominantly byte sequence-matching white-box scanners premised on classic TM function-based assumptions, we suggest moving towards more 'Interactive Computation' through the use of interactive iterative games and black-box process modeling.

6 Challenges ahead

So far, PQS models have to be provided a priori and cannot be generated on the fly; at most, parameters can be tuned. Dynamic generation of such models that accurately reproduce the malware's hypothesis structure is a formidable hurdle. Fruitful research directions may include genetic programming, and recombinant mechanisms that may resemble the very same metamorphic engines that produced the malware. From the internal PQS models, a novel *process-based malware taxonomy* will have to arise. The description will be richer, more holistic than traditional behaviorial profiles in that the malware's internal hypothesis structure (its decision loop and control flows - traditionally the domain of white-box analysis) will have to largely be inferred by the effects of strategies used in the iterative games.

This taxonomy, together with a suitable process-oriented visual language, visualization tools, and emulation environments, will have to be developed *sui generis* for the study of these engineered phenomena. We proposed here a systems-oriented framework which leverages techniques from Interactive Computations, Bayesian statistics, iterative 2-player (possibly n-player) imperfect non zero-sum games, and process query analysis.

We end on a philosophical note. Tackling this intriguing problem of approaching, comprehending and cataloguing complex dissimulated entities interactively from the outside, we may find that this reproduces - in the context of digital biotopes - in spirit the *naturalist approaches* of Alexander v. Humboldt/E.O Wilson/Rachel Carson. The faint outlines and first touches of this program are found in [66, 46, 67, 16, 47, 68, 69]. In effect, these directions may be instrumental in laying the groundwork for a new subfield of the natural sciences.

References

1. Szor, P. [3] 269–293
2. Christodorescu, M., Jha, S.: Static analysis of executables to detect malicious patterns. In: Security '03:Proceedings of the 12^{th} USENIX Security Symposium, USENIX Association, USENIX Association (August 2003) 169–186
3. Szor, P.: The Art of Computer Virus Research and Defense. Addison-Wesley Professional, Upper Saddle River (NJ) (February 2005)
4. Filiol, E.: Metamorphism, formal grammars and undecidable code mutation. International Journal of Computer Science **2**(2) (2007) 70–75
5. Filiol, E.: Formalisation and implementation aspects of *k*-ary (malicious) codes. Journal in Computer Virology **3**(2) (2007) 75–86
6. Computer Economics: 2007 malware report: The economic impact of viruses, spyware, adware, botnets, and other malicious code. Technical report, Computer Economics Inc. (2007) accessed Oct. 17^{th}, 2007.
7. Internet System Consortium: Internet domain survey host count. Technical report, Internet System Consortium (July 2007) accessed Oct. 17^{th}, 2007.
8. Clementi, A.: Anti-virus comparative no. 13. Technical report, Kompetenzzentrum IT, Insbruck (Austria) (Februray 2007) `http://www.av-comparatives.org/seiten/ergebnisse/report13.pdf`.
9. Wong, W., Stamp, M.: Hunting for metamorphic engines. Journal in Computer Virology **2**(3) (December 2006) 211–229
10. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying file types by n-gram analysis. In: SMC '05:Proceedings from the Sixth Annual IEEE Information Assurance Workshop on Systems, Man and Cybernetics, West Point (NY) (June 2005) 64– 71
11. Weber, M., Schmid, M., Schatz, M., Geyer, D.: A toolkit for detecting and analyzing malicious software. In: ACSAC '02: Proceedings of the 18^{th} Annual Computer Security Applications Conference, Washington (DC) (2002)
12. Emm, D.: AV is alive and well. Virus Bulletin (September 2007) 2
13. Zamboni, D., Krügel, C., eds.: Recent Advances in Intrusion Detection, 9th International Symposium, RAID 2006, Hamburg, Germany, September 20-22, 2006, Proceedings. In Zamboni, D., Krügel, C., eds.: RAID. Volume 4219 of Lecture Notes in Computer Science., Springer (2006)

14. Newsome, J., Karp, B., Song, D.X.: Paragraph: Thwarting signature learning by training maliciously. [13] 81–105
15. Walenstein, A., Mathur, R., Chouchane, M.R., Lakhotia, A.: Normalizing metamorphic malware using term rewriting. In: SCAM '06: Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'06), Washington, DC, IEEE Computer Society (2006) 75–84
16. Chouchane, M.R., Lakhotia, A.: Using engine signature to detect metamorphic malware. In: WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode, New York, NY, ACM Press (2006) 73–78
17. Finones, R., Fernandez, R.: Solving the metamorphic puzzle. Virus Bulletin (March 2006)
18. Cohen, F.: Computer Viruses. PhD thesis, University of Southern California (January 1986)
19. Chess, D., White, S.: An undetectable computer virus. In: VB '00: Proceedings of the 2000 Virus Bulletin Conference. (September 2000)
20. Spinellis, D.: Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory **49**(1) (January 2003) 280–284
21. Wegner, P., Goldin, D.: Principles of problem solving. Commun. ACM **49**(7) (2006) 27–29
22. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society **2**(42) (1936) 230–265
23. Zittrain, J.: The generative internet. Harvard Law Review **119**(7) (May 2006) 1974–2040
24. Li, L., Alderson, D., Willinger, W., Doyle, J.: A first-principles approach to understanding the internet's router-level topology. In: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, New York (NY), ACM Press (2004) 3–14
25. Carlson, J.M., Doyle, J.: Highly optimized tolerance: A mechanism for power laws in designed systems. Physical Review E **60**(2) (1999) 1412+
26. Barry, J.: The story of influenza. In Knobler S, Mack A, M.A.L.S., ed.: The Threat of Pandemic Influenza: Are We Ready? The National Academies Press (2005) 58
27. Quarterman, J.S.: Monoculture considered harmful. First Monday **7**(2) (2002)
28. Bailey, M.G.: Malware resistant networking using system diversity. In: SIGITE '05: Proceedings of the 6th conference on Information technology education, New York, NY, ACM Press (2005) 191–197
29. Moore, D., Shannon, C., k claffy: Code-Red: a case study on the spread and victims of an internet worm. In: IMW '02: Proceedings of the 2^{nd} ACM SIGCOMM Workshop on Internet Measurement, New York, NY, ACM Press (2002) 273–284
30. Forrest, S., Somayaji, A., Ackley, D.: Building diverse computer systems. In: HOTOS '97: Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI), Washington, DC, IEEE Computer Society (1997) 67
31. Zhang, Y., Vin, H., Alvisi, L., Lee, W., Dao, S.K.: Heterogeneous networking: a new survivability paradigm. In: NSPW '01: Proceedings of the 2001 workshop on New security paradigms, New York, NY, ACM Press (2001) 33–39
32. Geer, D.: Cyberinsecurity: The cost of monopoly. Technical report, Computer and Communications Industry Association (September 2003)
33. Survey, N. `http://news.netcraft.com/` accessed Feb.7^{th}, 2007.
34. Wheeler, D.A.: Why open source software. `http://www.dwheeler.com/oss_fs_why.html#market_share` (November 2005) accessed Feb. 7^{th}, 2007.

35. Todar, K.: Growth of bacterial populations. http://textbookofbacteriology.net/growth.html (2002) U Wis. (Madison) Dep. of Bacteriology.
36. Gingerich, P.: Quantification and comparison of evolutionary rates. American Journal of Science **293A** (1993) 453–478
37. Kraus, J.: Selbstreproduktion bei Programmen. Master's thesis, Universität Dortmund (Germany) (February 1980)
38. Peikari, C.: Analyzing the crossover virus: The first pc to windows handheld cross-infector. http://www.informit.com/articles/printerfriendly.aspx?p=458169 (March 2006) accessed Oct. 17^{th}, 2007.
39. Pastor-Satorras, R., Vespignani, A.: Epidemic spreading in scale-free networks. Phys. Rev. Lett. **86**(14) (Apr 2001) 3200–3203
40. Inc, C.: Malware outbreak trend report: Bagle-worm. Technical report (March 2007) accessed Oct. 17^{th}, 2007.
41. Inc, C.: Server-side polymorphic viruses surge past av defenses. Technical report (May 2007) accessed Oct. 17^{th}, 2007.
42. Schechtman, G.M.: Manipulating the ooda loop: The overlooked role of information resource management in information warfare. Master's thesis, Air Force Institute of Technology (1996)
43. Ullman, D.G.: "OO-OO-OO!" the sound of a broken OODA loop. Journal of Defense Software Engineering **20**(4) (April 2007) 22–25
44. Kullback, S., Leibler, R.A.: On information and sufficiency. The Annals of Mathematical Statistics **22**(1) (March 1951) 79–86
45. Hoepers, C., Steding-Jessen, K., Montes, A.: Honeynets Applied to the CSIRT Scenario. In: Proceedings of the 15th Annual Computer Security Incident Handling Conference, Ottawa, Canada (June 2003)
46. Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The Nepenthes Platform: An Efficient Approach to Collect Malware. [13] 165–184
47. Crandall, J.R., Su, Z., Wu, S.F., Chong, F.T.: On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In: CCS '05: Proceedings of the 12th ACM conference on Computer and communications security, New York, NY, ACM Press (2005) 235–248
48. Eagle, C.: Ripple in the gene pool. In: Defcon 14, Las Vegas (NV) (July 2006)
49. Shacham, H., Page, M., Pfaff, B., Goh, E.J., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: CCS '04: Proceedings of the 11th ACM conference on Computer and communications security, New York, NY, ACM Press (2004) 298–307
50. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. [13] 226–248
51. Bilar, D.: On callgraphs and generative mechanisms. Journal in Computer Virology **3**(4) (November 2007)
52. George Cybenko, Vincent H. Berk, V.C.R.S.G., Jiang, G.: An overview of process query systems. In: Proceedings of the SPIE: Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense. Volume 5403. (2004)
53. Roblee, C., Cybenko, G.: Implementing large-scale autonomic server monitoring using process query systems. In: ICAC '05: Proceedings of the Second International Conference on Automatic Computing, Washington, DC, IEEE Computer Society (2005) 123–133
54. Cybenko, G.: The mathematics and algorithmics of process detection. Institute for Pure and Applied Mathematics (UCLA) (7 2005)

55. Goldin, D., Wegner, P.: The church-turing thesis: Breaking the myth. Lecture Notes in Computer Science: New Computational Paradigms (2005) 152–168
56. Rado, T.: On non-computable functions. Bell Systems Technical Journal **41**(3) (May 1962) 877–884
57. Goldin, D.Q., Smolka, S.A., Wegner, P.: Interactive Computation: The New Paradigm. Springer (September 2006)
58. Binmore, K.: Playing for real: a text on game theory. Oxford University Press (2007)
59. Shor, M.: Notes on game theory. `http://www.gametheory.net/lectures/` (2007)
60. Liston, T.: Labrea. `http://www.hackbusters.net` (2003)
61. Borders, K., Zhao, X., Prakash, A.: Siren: Catching evasive malware (short paper). In: SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy, Washington, DC, IEEE Computer Society (2006) 78–85
62. Alpcan, T., Basar, T.: A game theoretic approach to decision and analysis in network intrusion detection. In: Proceedings of the 42^{nd} IEEE Conference on Decision and Control, Maui (HI) (December 2003)
63. Liu, Y., Comaniciu, C., Man, H.: A bayesian game approach for intrusion detection in wireless ad hoc networks. In: GameNets '06: Proceeding from the 2006 workshop on Game theory for communications and networks, New York, NY, ACM Press (2006) 4
64. von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. Commun. ACM **47**(2) (February 2004) 56–60
65. NEC: SEAL VoIP anti-spam. `http://www.itwire.com.au/content/view/8981/127/` (accessed Feb. 11^{th},2007)
66. Wasenaar, T., Blaser, M.: Contagion on the internet. Emerging Infectious Diseases **8**(3) (March 2002) 335–336
67. Hall, K.J.: Thwarting Network Stealth Worms in Networks Through Biological Epidemiology. PhD thesis, Virginia Polytechnic Institute and State University (May 2006)
68. Crandall, J.R., Wassermann, G., de Oliveira, D.A.S., Su, Z., Wu, S.F., Chong, F.T.: Temporal search: detecting hidden malware timebombs with virtual machines. In: ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, New York, NY, ACM Press (2006) 25–36
69. Zou, C.C., Gong, W., Towsley, D.: Worm propagation modeling and analysis under dynamic quarantine defense. In: WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode, New York, NY, ACM Press (2003) 51–60