Toward a Mathematical Understanding of the Malware Problem

Michael Stephen Fiske

Aemea Institute. San Francisco, CA. **

Malware exploits a weakness in current computer systems: user authentication [1] does not guarantee the execution of the user's intended action (e.g., authorization of a valid bank transaction), even when the authentication is tightly integrated with strong cryptographic protocols. As aptly stated by Adi Shamir, *cryptography is typically bypassed, not penetrated* [2]. In practical terms, this weakness creates vulnerabilities in computer systems that operate critical infrastructure such as the air traffic control system [3] and GPS [4]. In recent years, malware played a significant role in cyberattacks. In 2011, the RSA SecurID breach [5] compromised the computer networks of more than 700 institutions even though the adversary did not possess a valid SecurID device. A malware exploit in Adobe Flash helped launch this compromise. In a distinct cyberattack, Stuxnet [6] further exposed these vulnerabilities, even though the processors weren't executing an operating system. The programmable logic controller in the centrifuges was reverse engineered. Malware was planted in the controller so that the centrifuge speed was erratically increased and decreased, causing vibrations, while indicating that the centrifuge was operating properly. These "malware" vibrations disabled some of the centrifuges.

Over a 20 year study, the DARPA program, CRASH [7], compared the number of lines of source code in security software versus malware. Security code grew from about 10,000 to 10 million lines; while malware was almost constant at 125 lines. This study found that an almost constant number of lines of malware code can hijack a computer program, independent of the security software's size and complexity. It seems unlikely that detection methods [8,9,10] can provide an adequate solution to the malware problem. First, it is known that there is no Turing machine algorithm that can detect all malware [11]. Second, some recent malware implementations use NP problems [12] to encrypt and hide the malware [13]. Detection methods are not only up against fundamental limits in theoretical computer science [14], but also malware, in practice, uses the same tools (i.e., cryptography and camouflage) as white hats. In 2013, the New York Times network was hacked; anti-virus software was unable to detect this malware, even though it was on the network for many months [15,16].

Rather than continue to pursue detection, our research explains a register machine's [17,18] susceptibility to malware as a consequence of the structural instability of conventional computation. The register machine executes instructions one-at-a-time. Programming languages such as `C`, `Java`, `Lisp` and `Python` – that are Turing complete – depend upon branching instructions. While conditional branching instructions are not required for universal computation, Rojas's

** Email: `mf@aemea.org`

methods [19] still use unconditional branching and program self-modification. Further, figure A.14 in [18] shows that over 75% of the control flow instructions, executed on conventional processors, are conditional branch instructions. After a branching instruction of a register machine program has been sabotaged, even if there is a routine to check if the program is behaving properly, this friendly routine may never get executed. *The sequential execution of register machine instructions cripples the program from protecting itself.*

In contrast to substantially altering computing behavior by changing only one memory address, the Lashley experiments [20] demonstrated that rats were still able to navigate a maze, even though considerable portions of cortical tissue were removed in various locations of their cortex.

To the best of the author's knowledge, prior research [11,21,22,23] has not attempted to understand malware susceptibility in terms of structural stability. Stability has been studied extensively in dynamical systems [24,25,26,27,28,29]. Our research shows how the computation of a Turing machine corresponds to the iteration of a dynamical system, composed of a finite set of affine maps in the x-y plane. This correspondence induces a metric and demonstrates a structural instability in a Universal Turing machine encoding. In practice, the instability of conventional computation enables malware authors to sabotage the behavior of a computer program, by making only small changes to the original, uninfected program. One research direction proposes to better understand instability in conventional computation; a second direction suggests a search for stable computation in unconventional machines such as the *active element machine* [30,31,32].

References

1. Keith Mayes and Konstantinos Markantonakis (editors). Smart Cards, Tokens, Security and Applications. Springer, 2008.
2. Adi Shamir. Cryptography: State of the Science. ACM. Turing Award Lecture. June 8, 2003. `http://amturing.acm.org/vp/shamir_2327856.cfm`.
3. Michael Nolan. Fundamentals of Air Traffic Control. 5th edition, Cengage Learning, 2010.
4. Pratap Misra and Per Enge. Global Positioning System: Signals, Measurements, and Performance. Revised 2nd Edition, Ganga-Jamuna Press, 2011.
5. *SecurID — Wikipedia*. `http://en.wikipedia.org/wiki/RSA_SecurID`.
6. David Kushner. The Real Story of Stuxnet. IEEE Spectrum, Feb. 26, 2013.
7. Cheryl Pellerin. DARPA Goal for Cybersecurity: Change the Game. American Forces Press Service, December 20, 2010.
8. John Mitchell and Elizabeth Stillson. Detection of Malicious Programs. U.S. Patent 7,870,610, 2011.
9. Diego Zamboni (editor). Proc. of the 5th Intl. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. LNCS. Springer. July 2008.
10. A. Moser, C. Kruegel and E. Kirda. Limits of Static Analysis for Malware Detection. IEEE. 23rd Annual Computer Security Applications Conf., 2007.

11. Fred Cohen. Computer Viruses Theory and Experiments. Computers and Security, **6**(1) 22–35, Feb. 1987. `http://all.net/books/Dissertation.pdf`.
12. Stephen Cook. THE P VERSUS NP PROBLEM. Clay Math Institute, 2013. Official Problem Description. `http://www.aemea.org/math/P_vs_NP.pdf`
13. Eric Filiol. Malicious Cryptology and Mathematics. Cryptography and Security in Computing. Chapter 2. Intech, March 7, 2012.
14. Eric Filiol. Computer viruses: from theory to applications. Springer, 2005.
15. Nicole Perlroth. Hackers in China Attacked The Times for Last 4 Months. New York Times, January 30, 2013.
16. Gerry Smith. Why Antivirus Software Didn't Save The New York Times From Hackers. Huffington Post, January 31, 2013.
17. Harold Abelson and Gerald Jay Sussman with Julie Sussman. Structure and Interpretation of Computer Programs. Second Edition, MIT Press, 1996.
18. John Hennessy and David Patterson. Computer Architecture. 5th Edition, Morgan Kaufmann, 2012.
19. Raul Rojas. Conditional Branching is not Necessary for Universal Computation in von Neumann Computers. Journal of Universal Computer Science. **2**, No. 11, 756–768, 1996.
20. Karl Spencer Lashley. Studies of cerebral function in learning XII. Loss of the maze habit after occipital lesions in blind rats. Journal of Comparative Neurology. **79**, Issue 3, 431–462, December, 1943.
21. Len Adleman. An Abstract Theory of Computer Viruses. Advances in Cryptology – CRYPTO 2008. LNCS **403**, Springer, 1988.
22. Fred Cohen. Computational Aspects of Computer Viruses. Computers and Security, **8**(4) 325–344, June 1989.
23. G. Bonfante, M. Kaczmarek, and J.-Y. Marion. On Abstract Computer Virology from a Recursion-theoretic Perspective. Journal in Computer Virology. **1**, No. 3-4, 2006.
24. A. Andronov and L. Pontrjagin. Systmes Grossiers. Dokl. Akad. Nauk., SSSR, **14**, 247–251, 1937.
25. D. Anosov. Geodesic flows on closed Riemannian manifolds of negative curvature. Proc. Steklov. Inst. Math. **90**, 1967.
26. J. Palis and S. Smale. Structural Stability Theorems. Proc. Symp. Pure Math. AMS. **14**, 223–232, 1970.
27. John Franks. Necessary Conditions for Stability of Diffeomorphisms. Trans. Amer. Math. Soc. **158**, No. 2, 301–308, 1971.
28. Clark Robinson. Structural Stability of C^1 Diffeomorphisms. Journal of Differential Equations. **22**, 28–73, 1976.
29. Keonhee Lee and Kazuhiro Sakai. Structural stability of vector fields with shadowing. Journal of Differential Equations. **232**, 303–313, 2007.
30. Michael S. Fiske. The Active Element Machine. Proc. of Comp. Intelligence. Autonomous Systems: Developments and Trends. **391**, 69–96, Springer, 2011.
31. Michael S. Fiske. Turing Incomputable Computation. Turing-100 Proceedings. Alan Turing Centenary. **10**, 69–91, EasyChair, 2012. `http://www.aemea.org/Turing100`.
32. Michael S. Fiske. Quantum Random Active Element Machine. UCNC 2013 Proceedings. LNCS **7956**, 252–254, Springer, 2013. `http://www.aemea.org/UCNC2013`.