The Evolution of Fine-Grain Malware Behavior Analysis, From Static to Dynamic

C.W. Wang, and Shiuhpyng Shieh
NCTU
Email: ssp@cs.nctu.edu.tw

Conventional static analysis and detection of malware cannot hold the line due to evolutionary malware masquerading techniques, such as packing, polymorphism, and metamorphism. Hence, dynamic analysis techniques have become essential analysis tools for malware defense. Most dynamic analysis mechanisms try to execute the target in an isolated, monitored environment, such as virtual machine, to examine its behaviors thoroughly [1], [2]. Their effectiveness comes from the intrinsic characteristics of malware. To sabotage or to infiltrate computers, malware must alter the victim systems in an obscure way. No matter how minor these alterations are, they can always be detected and recognized using a fine-grained monitor. Dynamic information flow tracking is an outstanding example of such monitors. By running the target binary codes in an emulated (or virtualized) environment, each CPU instruction and memory access can be monitored and analyzed so that information flows, direct or indirect, can therefore be captured.

The real problem is performance. Dynamic analysis usually requires emulation causing serious performance degradation that is unacceptable for end users. Although advanced dynamic binary translation has provided significant improvement in emulation speed, the overall performance still cannot be comparable with real hardware due to the complex analysis procedures added to emulators. The computation overhead incurred by dynamic analysis of malware behaviors is often much more serious than the emulation itself. However, it is possible for the analysis workload to migrate to another CPU core in a multi-core system by decoupling emulation and analysis procedures [7], [8]. The primary challenge for tracking information propagation is to analyze CPU execution traces. Using a pipelined multithreading paradigm, emulation and analysis can be examined simultaneously.

Virtualization may be a brilliant solution to performance issues. Currently, nearly all commercial CPUs are equipped with virtualization techniques. One can easily create a virtualized environment to conduct dynamic analysis at an acceptable speed because of native execution. Unfortunately, the high performance does not come without a pitfall. Because CPU instructions are now executed natively, there are fewer chances for us to observe the system and direct its execution. Several possible techniques are being developed to help, such as demanded execution, or VMI (Virtual Machine Introspection) [3], [4]. Demanded execution switches back and forth between emulation and virtualization to benefit from the performance advantage provided by native execution without loss of flexibility for analysis. VMI is also an interesting topic when we analyze a virtual machine. To better understand what is happening inside the VM, we need VMI to transform collected, low-level data to meaningful information of higher abstraction levels such as processes and files.

Cloud computing can also plays an important role in malware analysis. Along with the maturity of dynamic malware analysis and cloud computing techniques, it is possible to shift all the analysis tasks to a remote security service provider. Indeed, such services are already being developed and widely deployed. CWSandbox [5], and Anubis [6] are good examples of such online analyzing platforms. End users can submit unknown binary to these service providers, and expect the analysis result in a few minutes. There is a long way to go before such services become truly usable for real time analysis; it is however a potential research topic deserving our continuous attention and efforts.

[1] M. I. Sharif, W. Lee, W. Cui, and A. Lanzi, “Secure in-VM monitoring using hardware virtualization,” *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS, 2009.
[2] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. “Ether: Malware analysis via hardware virtualization extensions,” Proceedings of The 15th ACM Conference on Computer and

Communications Security. CCS 2008.
[3] J. Pfoh, C. Schneider, and C. Eckert, “A formal model for virtual machine introspection,” *Proceedings of the 1st ACM Workshop on Virtual Machine Security*. VMSec 2009.
[4] P. Moret, W. Binder, and A. Villazon, “CCCP: complete calling context profiling in virtual execution environments,” *Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation* . PEPM 2009.
[5] Carsten Willems, Thorsten Holz, Felix Freiling, “Toward Automated Dynamic Malware Analysis Using CWSandbox,” *IEEE Security and Privacy, vol. 5, no. 2, pp. 32-39,* Mar./Apr. 2007.
[6] U. Bayer, C. Kruegel, and E. Kirda, “TTAnalyze: A Tool for Analyzing Malware,” *15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference*, April 2006.
[7] O. Ruwase, P. B. Gibbons, T. C. Mowry, V. Ramachandran, S. Chen, M. Kozuch, and M. Ryan, “Parallelizing dynamic information flow tracking,” *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*. SPAA 2008.
[8] J. C. Martinez Santos, Y. Fei, and Z. J. Shi, “PIFT: efficient dynamic information flow tracking using secure page allocation,” *Proceedings of the 4th Workshop on Embedded Systems Security*. WESS 2009.