Avum Cyber Security Approach: Implementing Java Pathfinder (JPF) Technologies in Modern Cyber Security Solutions

Dustin Kramer, *Research Engineer, Avum, Inc.* and Jerry Hill, *Senior Science Advisor, Avum, Inc.*

Abstract—If you believe that your system has not been intruded upon and your private information compromised, you could be right. But, with the existence of pervasive, ever-advancing cyber threats, it is equally likely that you are wrong. Avum has been researching and implementing modern, efficient cyber security methods to attempt to eradicate this issue altogether (see white paper *Avum Cyber Security Approach: Prevention, Detection, and Threat Removal*). This paper outlines a unique method researchers at Avum have been developing that combines current cyber security methodologies with JPF technology that can potentially achieve this goal.

1 Introduction

AVUM, Inc. has been active in Cyber Security on our own behalf and that of our commercial and DoD customers for several years. Thus, our customers application domains present a logical path for us in expanding research already begun in developing a Cyber Security application to detect and prevent Cyber intrusion into target systems. This document provides an overview for our plan to develop techniques and tools that utilize the methods and technologies of Java Pathfinder (JPF) to run in a Virtual Machine (VM) environment. These tools would essentially emulate JPFs advanced Java^{TM} bytecode model checking and path execution capabilities to analyze all applications prior to their being allowed to run in operational computer systems. The goal of this approach would be the ability to be able to recognize malware threats by safely examining them in a virtual environment, prior to their intrusion into one's system.

2 Background: What is JPF?

For those who are not familiar with Java Pathfinder, JPF is a system, developed at the NASA Ames Research Center, to verify executable Java^{TM} bytecode programs. The core of JPF is a Virtual Machine (VM) for Java^{TM} bytecode, which means it is a "program which you give Java programs to execute." It is thus, essentially, a "VM running on top of a VM" that is used to find defects in programs. One supplies JPF with properties to check for, and JPF returns with a report that says if the properties hold (NASA Ames Research Center, 2007).

A key aspect of the JPF technology lies in its ability to check all possible execution choices of a program for the defined and inputted defects. JPF can "identify points in your program from where execution could proceed differently, then systematically explore all of them. This means JPF (theoretically) executes all paths through your program, not just one like a normal VM." What JPF then does if it finds a defect that matches the given properties, is it will provide the complete execution history leading up to the defect to easily allow the user to see what caused it. It may seem unfeasible for JPF to be able to execute all potential program paths given that the number of these execution choices may increase exponentially. This is why JPF employs "state matching" and can restore previous program states if it has reached one it has already seen (NASA Ames Research Center, 2007).

Depending on how the user configures JPF, it is capable of detecting a wide range of program defects, including ones as simple as deadlocks and unhandled exceptions (known as "non-functional" properties that no Java program should violate). But the beauty of JPF lies in its ability to check against your own defined properties, and this is where JPF technology can be utilized in cyber intrusion detection and protection (NASA Ames Research Center, 2007).

3 The Current Problem

Intrusive software proliferation is increasingly sophisticated. Intruders malicious software, commonly referred to as "malware," is defined as software with malicious intent. That is, a program, or programs, designed to mine data that can be used for financial gain at the owners expense, or inflict harm on the host system in which it executes or the network over which it communicates. Current malware can also be extremely difficult to detect and can be easily and unintentionally allowed into a system. Under certain circumstances, one could even be completely unaware that the malware is being housed by the host, having been unsuspectingly installed by even the system's owner or owner's representative.

Increasingly complex computer systems provide a rich environment for malware attacks, or a malware host that surreptitiously gleans proprietary information from unwary applications. The larger and more complex a system, the more difficult malware detection becomes - or corresponding proof of its absence. The threat of malware attacks, therefore, presents an unavoidable computer security risk, making protection from, and/or detection of the presence of malicious code critical to systems operating on, and transmitting and storing proprietary or classified information.

4 A Proposed Solution

4.1 Overview

Of course it would be the ultimate achievement to prevent malware intrusion into the system in the first place. Therefore,

- *D. Kramer is a Research Engineer at Avum, Inc., playing an integral role on the company's Research and Development team. E-mail: dustinkramer@avum.com*
- *J. Hill is responsible for Avum's Program Execution and Life-Cycle Engineering. E-mail: jhill@avum.com*

Manuscript created December 11, 2013.

Avum's approach is to achieve that goal. The plan is to intercept software prior to its being loaded into the target system, including complex systems with numerous disparate applications with the secure network infrastructure and servers, utilizing JPF's advanced technologies. We at Avum are developing a prototype system and are pursuing research leading to a yet more sophisticated system to run applications in a virtual environment prior to being loaded into the target system. Utilizing JPF, every possible program path would be analyzed in the virtual environment, with the composite results being assessed against properties that we would expect to find only in malicious software (see Fig. 1 below). This analysis would further result in the identification of new properties to expand the existing malware properties list to be used in future path analysis. Essentially, prior to its introduction into your system, a program enters the JPF-based program evaluator, where it has all of its program paths executed and checked for signs of malware. Anything recognized as potentially hazardous is then outputted by the malware detection system. Although JPF is designed for use with Java^{TM} programs, the ultimate goal would be to extend this technology to understand other language platforms as well.

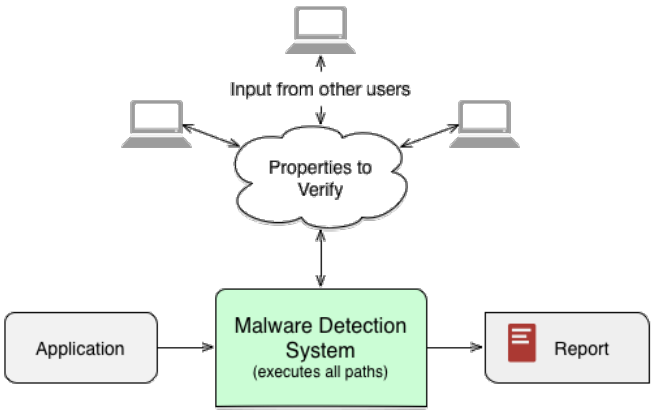

Fig. 1: Malware Detection System Concept

A key component, then, in the development of a JPF-based malware detection agent, is the development of a comprehensive list of malware properties. For this method to be effective, a detailed understanding of the structure and form of modern malware would be required, as for the JPF system to work, malicious software properties need to be entered in order to be assessed against. To address this issue, the developed platform would be designed to "learn" over time. That is, an initial list of known malware properties would be provided, then the platform would cooperate in a network of users of these systems to share information on malware to help the virtual machine to learn about existing threats even before it encounters them (again, see Fig. 1 above). The construction of this initial list, however, could be quite difficult, as malware and their developers are becoming incredibly adaptive and are finding new ways to intrude upon systems. This is an area that Avum has been focusing research in, but would still greatly benefit from further research.

4.2 Initial Malware Properties

We must also assume that a malware developer might anticipate any given approach, and take his or her own countermeasures. We could thus include counter-countermeasures in our original plan. For example, if an application to be loaded on the target system contained dead code, we recognize that the path analyzer would not find it. The only way malicious software could reach the dead code to fulfill its malicious intent would be to modify itself during execution to provide a path to that code. Therefore, self-modifying code might become part of the path analysis, thus a property to be checked for when analyzing suspicious programs.

Once having found the path to the dead code, it would be included in the path analysis. Thus, self-modifying code and dead code would represent two properties in our original properties list. Our initial research, currently in progress, is to determine as many of these properties as possible to be applied in a virtual environment. Through the analysis of a large number of applications, the virtual system would "learn" an extensible number of characteristics of malware, using that information to dynamically expand the properties list, with a corresponding increase in the malware detection rate.

Since current firewalls are notoriously ineffective and are routinely being defeated by malware developers, current malware detection focuses on catching malware in the act after it has already invaded the target system. Unfortunately, no matter how good detection is, once the malware is running on the target platform the risk is significant, with the potential for damage then being measured in seconds, if not milliseconds.

4.3 Limits of Current Malware Detection

Our investigation finds that even the best of the malware detection applications advertise detection only minutes after initial intrusion, but a lag of even seconds could be too late to stop the transfer of critical information to external hostile destinations, or significant damage to owner applications and databases. Thus, the best malware protection would be to not allow the malware into the owner platform in the first place. This level of security is not currently being provided industry-wide to an extent that would provide a comfortable level of confidence in Information Assurance (IA) in typical system environments.

The current approach is shackled with severe limitations. Techniques now being employed, of which we are aware, are largely based on the assumption that malware invasion is inevitable, with detection being in the host system as soon as possible on day one of the intrusion, followed by quarantine and elimination of the threat. Detection techniques that are signature-based, anomaly-based, or that provide detection based on behavior are run on the application only after it is running in the system.

The two most common forms of malware detection are currently signature-based and anomaly-based detection, which require pre-knowledge of that particular malware, and are only initiated after the malware has invaded a target system. With no pre-knowledge, malware can go undetected for days, if not weeks or months, inflicting significant damage and potentially transferring confidential, if not highly classified information to the malware source. The limitations of current methods of malware detection on the host computer, at best, briefly allow access to databases and applications, and at worst can allow malware to execute for months without detection. Thus, we treat allowing malware access to the host system as being unacceptable. The basic premise of our developing plan is that malware must be detected and eliminated prior to intrusion.

4.4 Future Plans

Avum's investigation is ongoing. We will continue our investigation into malicious software detection, starting with the currently understood malware obfuscation and signature-based properties, progressing to development of predictive capabilities to recognize obfuscation metamorphoses of existing malware, and potential new malware, which may, or may not be related to known malware patterns or their metamorphic progeny. This phase will provide properties to be applied during application path analysis in the VM environment.

We envision a virtual system to detect recognizable system execution that would signal unusual activity attributable to the presence of malware. With malware being detected, the system administrator would be given an alarm, so that action could be taken to extract as much information as possible about the malware then remove and isolate it from the system for further study. Post-intrusion detection still has significant value in detecting malware making it through the virtual system. We would facilitate information exchange between the virtual system and the imbedded malware detection applications to ensure the latest properties are available to both.

Our plan requires an initial list of properties, which will be established to provide a baseline for our pre-intrusion malware protection. This properties list will originally be based on known properties that can be gleaned from current desktop and laptop systems, and various mobile platforms, such as those in the Android-based mobile application, Nokia's Symbian, Apple's iOS, Microsoft's Windows Phone 7, RIM's BlackBerry OS, and embedded Linux distributions such as Maemo and MeeGo marketplaces. To ignore these mobile applications would be to deny the rapidly accelerating forward momentum of technology. Such an approach is necessary to place us one step ahead, rather than one step behind, the creators and maintainers of malicious software. We would cooperate in a network of users of these systems to share information on malware to help them and us to predict malware prior to its arrival on any system, whether desktop or mobile.

Avum views this challenge as being one of advancing the state-of-the-art in malware detection, rather than improving detection by observing syntactic patterns or anomalous behavior, both of which are subject to time lags or unacceptable uncertainty in detection results. As demonstrated in this paper, we are already researching the potential of developing a semantics-based detection framework that recognizes malicious code through studying the semantics, rather than the signature of a malicious program. That is, by monitoring behavior of known malware, we could establish a pattern of behavior that must break with the system architecture it was meant to invade and compromise.

It is also important to note that our VM analysis is application naïve, and would thus provide malware detection as a front end to any system based on system-specific properties. Therefore any system, stationary or mobile, requiring cyber security should benefit from our VM malware detection.

5 Conclusion

There is no present solution available that completely combats the threat posed by current pervasive and dynamic cyber threats. We at Avum feel that we are on the right path toward developing a malware detection system that could eliminate this risk for commercial businesses, high-level government agencies, and home users. We are avidly continuing our research in the development of a detection system that utilizes Java Pathfinder technology because it has the potential to revolutionize cyber security approaches. As our malware detection technique matures through "learning" from malware inspection by applying comprehensive program path analysis, we can eventually eradicate malware intrusion altogether.

Acknowledgments

Mr. Hill would like to acknowledge his maternal grandfather, Harold S. Avery who instilled in him a love for studying our planet, our solar system, and the vastness of the Universe; and his two brothers, Jerry's uncles, Dr. George Avery former proprietor of the Brooklyn Botanical Gardens, and Dr. Robert Avery, Electrical Engineer at Bell Labs, Naperville, IL, for engendering in him an unending and unquenchable thirst for exploring the unknown. Mr. Kramer would like to acknowledge his parents, Jeffrey and Quinton Kramer, for always being there to provide support, advice, and love; and his sister Katie, for being a role-model in learning how to live a passionate and full life.

References

[1] NASA Ames Research Center. *What is JPF?* October 30, 2007. <http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/what_is_jpf>

[2] The Department of Homeland Security. *Cyber Security.* 2013. <https://www.dhs.gov/topic/cybersecurity>

[3] Committee on Enhancing the Robustness and Resilience of Future Electrical Transmission and Distribution in the United States to Terrorist Attack, Board on Energy and Environmental Systems, Division on Engineering and Physical Sciences. *Terrorism and the Electric Power Delivery System.* The National Academic Press. 2012.

[4] The Economist. *Cyber War: The Worm Turns.* December 4, 2008. <http://www.economist.com/node/12725712>

[5] Wikipedia.org. *Struxnet*. 2013. <http://en.wikipedia.org/wiki/Struxnet>

Dustin Kramer Mr. Kramer is a Research Engineer at Avum, Inc., performing in-depth research, implementation, and analysis for many of Avum's engineering projects. With a degree in Mathematics from UC Berkeley and having performed extensive amounts of research across a broad spectrum of topics, including Pure Mathematics, Physics and Human Psychology, Dustin is well-equipped to provide a comprehensive and unique perspective when approaching problems and developing solutions. Mr. Kramer is currently working to develop service-based Intelligent Agents (IA) to extract meaning and relevance from data through fusion across heterogeneous data sources.

Jerry Hill Mr. Hill, as Senior Science Advisor for Avum, Inc., brings to the company his 40 years of experience in computer system architecture, design, and process-driven system development. He has worked in the capacity of System Architect, and Project Manager in the development of science-based, computer systems for NASA/JPL, Air Force, Navy, and Army contracts. He has proposed solutions for and won at least two contracts with each of these NASA and DoD clients. Mr. Hill is currently focusing on the acquisition of new business for the company, in interoperable Web-based applications, in which they excel.