Survey on Semantics-based Online Malware Detection

Yash Karwa[1] Pratiksha Rokade[2] Suraj Netke[3] Nikhil Bajad[4] Prof. Madhuri Hingane[5]
[1,2,3,4,5]B.E. Student
[1,2,3,4,5]Department of Computer Engineering
[1,2,3,4,5]Pune District Education Association College of Engineering, Pune, Maharashtra, India

Abstract— Recently, malware has more and more become an essential threat to embedded systems, whereas standard software package solutions like antivirus and patches haven't been so triple-crown in defensive the ever-evolving and advanced malicious programs. During this work, we tend to propose a hardware-enhanced design, GuardOL, to perform on-line malware detection. GuardOL could be a combined approach exploitation processor and FPGA. Our approach aims to capture the malicious behavior (i.e., high-level semantics) of malware. to the present finish, we tend to first propose the frequency-centric model for feature construction using system call patterns of renowned malware and benign samples. We then develop a machine learning approach (using multilayer perceptron) in FPGA to train classifier exploitation these options. At runtime, the trained classifier is employed to classify the unknown samples as malware or benign, with early prediction. The experimental results show that our answer can do high classification accuracy, quick detection, low power consumption and flexibility for simple practicality upgrade to adapt to new malware samples. One in every of the most benefits of our style is that the support of early prediction detecting 46th of malware at intervals first 30 minutes of their execution, while 97 of the samples at 100% of their execution, with < 3% false positives.

Key words: Malware Detection, Hardware-enhanced Architecture, Runtime Security, Early Prediction, Reconfigurable Malware Detection

I. INTRODUCTION

Nowadays anti-virus software system applications became essential to the everyday personal computer user. These trendy tools for computer malware scanning use a combination of algorithms to discover and prevent malicious software system from inflicting harm.

This project is concerning building a web anti-virus tool for scanning file systems and detecting malware. The project encompasses the two main areas of malware discovery techniques.

One approach that has been around since the inception of antivirus tools is that the signature-based detection that examines the file's key aspects for a known static fingerprint. The signature itself might be a sequence of bytes that represent the malicious code within the file or the cryptographic hash of the entire infected file. Once the program has access to the malware signature it goes through the suspected file and appears for a match. The present range of virus signatures is over a 1,000 and it's growing constantly. This can be why it's extremely necessary what string-matching rule is chosen.

The other and a lot of modern technique for virus detection is that the heuristic-based. A heuristic technique means that an approach to downside determination that's not sure to be optimum or excellent however enough. The heuristics-based detection approach depends on inspecting files for suspicious characteristics while not the assistance of signatures. As an example, the tool may check for malicious directions or junk code within the file. Another example for heuristic detection is death penalty the get into emulation surroundings and pursuit the behaviour. This specific instance of heuristic is termed file emulation meantime the previous is termed file analysis. The most distinction between the two is one focuses on the actions the file will once dead in an exceedingly controlled virtual surroundings whereas the opposite one focuses on the file itself. This can be the explanation that file analysis is additionally being known as static analysis and file emulation - dynamic analysis.

II. LITERATURE SURVEY

A. A Sense of Self for Unix Processes

This paper outlines an approach to intrusion detection that's quite completely different from alternative efforts so far, one that seems promising each in its simplicity and its utility. we've got planned a way for outlining self for privileged UNIX processes, in terms of traditional patterns of short sequences of system calls. they need shown that the definition is compact with relevance the house of possible sequences, that it clearly distinguishes between completely different styles of processes, and that it's hot and bothered by many completely different categories of anomalous, or foreign, behavior, permitting these anomalies to be detected. The ends up in this paper are preliminary, and there are attacks that our methodology isn't ready to notice. However, it's computationally economical to observe short-range orderings of system calls, and this method may doubtless offer the idea for anon-line computer system consisting of many detection mechanisms, some impressed by the human system, and others derived from cryptographic techniques and more traditional intrusion detection systems.

B. A virtual machine introspection based architecture for intrusion detection

They propose the thought of virtual machine contemplation, an approach to intrusion detection that co-locates an IDS on a similar machine because the host it's observance and lever-ages a virtual machine monitor to isolate the IDS from the monitored host. The activity of the host is analyzed by directly observant hardware state and inferring software package state supported a priori data of its structure. This approach permits the IDS to: maintain high visibility, provides high evasion resistance within the face of host compromise, provides high attack resistance because of sturdy isolation, and provides the distinctive capability to mediate access to host hardware, permitting hardware access management policies to be implemented within the face of total host compromise.

They showed that implementing our design is sensible and possible exploitation current technology by implementing an image VMI IDS and demonstrating its ability to discover real attacks with acceptable performance. They believe VMI IDS occupies a replacement and vital purpose within the house of intrusion detection architectures.

C. Dynamic analysis of malicious code

Malware analysis is that the method of decisive the aim and practicality of a given malware sample (such as a scourge, worm, or Trojan horse). This method could be a necessary step to be ready to develop effective detection techniques for malicious code. In addition, it's a vital necessity for the event of removal tools which will completely delete malware from an infected machine. Traditionally, malware analysis has been a manual method that's tedious and time-intensive. Sadly, the quantity of samples that require to be analyzed by security vendors on a day to day is consistently increasing. This clearly reveals the requirement for tools that automatize and alter components of the analysis method. In this paper, they gift TTAnalyze, a tool for dynamically analyzing the behavior of Windows executable. To the current finish, the binary is run in emulated software surroundings and its (security-relevant) actions are monitored. Particularly, they record the Windows native system calls and Windows API functions that the program invokes. One vital feature of our system is that it will not modify the program that it executes (e.g., through API decision swing or breakpoints), creating it tougher to find by malicious code. Also, our tool runs binaries in unmodified Windows surroundings, those results in wonderful emulation accuracy. These factors build TTAnalyze a perfect tool for quickly understanding the behavior of AN unknown malware.

D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software

In order to combat the speedy unfold of a replacement worm be- fore it will compromise an outsized range of machines, it is necessary to possess automatic attack detection and defense mechanisms. This paper we've got projected dynamic taint analysis and shown however it may be wont to notice and analyze most styles of software package exploits, while not requiring source code or special compilation of a program, so permitting it to simply be used on goods software package. It reliably detects man y attacks, and that we have found no false positives in our tests. Additionally, as a result of it monitors the execution of a program at a fine-grained level, TaintCheck may be won't to offer further info concerning the attack. It's presently able to determine the input that caused the exploit, show however the malicious input led to the exploit at a processor-instruction level, and determine the worth wont to write the protected information (e.g. the comeback address). Moreover, we've got shown that TaintCheck is especially helpful in an automatic signature generation system it may be used to modify linguistics analysis based mostly signature generation, enhance content pattern extraction based mostly sig-nature generation, and verify the standard of generated sig-natures.

III. PREVIOUS SYSTEM

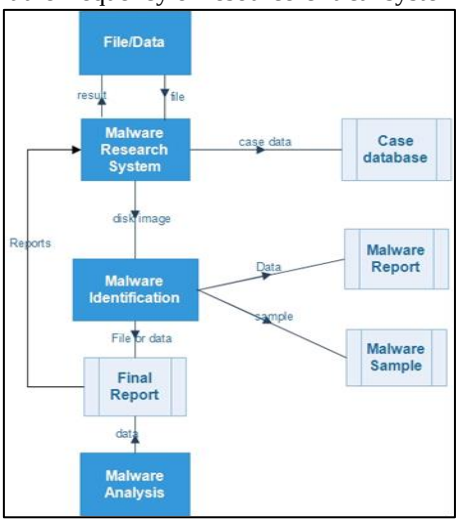
In order to combat the rapid spread of a new worm before it can compromise a large number of machines, it is necessary to have automatic attack detection and defense mechanisms. In this paper we have proposed dynamic taint analysis and shown how it can be used to detect and analyze most types of software exploits, without requiring source code or special compilation of a program, thus allowing it to easily be used on commodity software. It reliably detects many attacks, and we have found no false positives in our tests. In addition, because it monitors the execution of a program at a fine-grained level, TaintCheck can be used to provide additional information about the attack. It is currently able to identify the input that caused the exploit, show how the malicious input led to the exploit at a processor-instructionlevel, and identifythe value used to overwrite the protected data (e.g. the return address).

IV. PROPOSED SYSTEM

We propose a completely unique frequency-centralized feature construction model (FCM) to capture the linguistics of malware behavior. We tend to think about the frequency of resource important system concerns feature construction so as to discover malicious attacks that execute repetitive system calls to overload the system. Our planned model is predicated on the high-level linguistics of malicious behavior. We tend to adopt a dynamic approach to watch security-critical system calls together with their arguments and come back values to spot malicious mixtures through machine learning. we tend to aim at a generic illustration of a malicious behavior exploitation system calls so as to discover the variants of the malware, and even zero-day attacks that have similar linguistics. To the current finish, we tend to propose the subsequent rules to capture the high-level linguistics of malicious attacks.

Following process carried on files:

1) Group system calls in sets
2) Trace the memory mapping of file/device
3) Include the source file of the write operations
4) Count the frequency of resource-critical system calls



V. CONCLUSIONS

In this paper, we gave GuardOL, a hardware-enhanced design to notice malware at runtime. Our approach initial extracts the system calls and constructs the options supported the high level linguistics of malicious behavior. to the present finish, we have a tendency to propose a completely unique frequency-centralized model for feature construction. The options obtained from the benign and malware samples area unit then used for coaching the machine learning classifier, multilayer perceptron that is employed to notice the malware samples at runtime. The analysis results show that GuardOL is quick, effective and features a marginal performance overhead on the processor..

REFERENCES

[1] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in Proc. of S&P, 1996, pp. 120–128.
[2] T. Garfinkel, M. Rosenblum et al., "A virtual machine introspection based architecture for intrusion detection." in Proc. Of NDSS, vol. 3, 2003, pp. 191–206.
[3] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, "Dynamic analysis of malicious code," Journal in Computer Virology, vol. 2, no. 1, pp. 67–77, 2006.
[4] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," 2005.
[5] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in Proc. of ISEC, 2008, pp. 5–14.
[6] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in Proc. of ISSTA, 2012, pp. 122–132.
[7] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering." In Proc. of NDSS, 2009, pp. 8–11.
[8] M. Chandramohan, H. B. K. Tan, L. C. Briand, L. K. Shar, and B. M. Padmanabhuni, "A scalable approach for malware detection through bounded feature space behavior modeling," in Proc. of ASE, 2013, pp. 312–322.
[9] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario,"Towards an understanding of anti-virtualization and antidebugging behavior in modern malware," in Proc. of DSN, 2008, pp. 177–186.
[10] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode, "Rootkits on smart phones: attacks, implications and opportunities," in Proc. of HotMobile, 2010, pp. 49–54.
[11] X. Wang and R. Karri, "NumChecker: Detecting kernel controlflow modifying rootkits by using Hardware Performance Counters," in Proc. of DAC, 2013, pp. 1–7.

