International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 02– Issue 06, November 2013

Malware Detection from a Virtual Machine

Correlating Unusual Keystrokes, Network Traffic, and Suspicious Registry Access

Nathaniel Amsden
Department of Computer Science
Sam Houston State University
Huntsville, TX, USA

Cihan Varol
Department of Computer Science
Sam Houston State University
Huntsville, TX, USA
Email: cxv007 {at} shsu.edu

Abstract—**Current anti-virus malware detection methods focus on signature-based methods. Recent research has introduced new, effective methods of malware detection. First, recent research including cloud-based monitoring and analysis, joint network-host based methods, feature ranking, machine learning and kernel data structure invariant monitoring are reviewed. Second, virtual machine based malware detection is proposed. This method combines network traffic analysis through keystroke analysis and registry anomaly detection to detect malware. It correlates suspicious network activity with suspicious registry accesses in order to detect malware with a higher confidence and lower false positives.**

Keywords-keystroke analysis; malware detection; registry analysis; traffic analysis; virtual machine

I. INTRODUCTION

Current home and corporate malware detection primarily focus on anti-virus signature-based methods. Recent research in the field of malware detection has introduced new methods such as cloud-based analysis, machine learning, joint network-host methods and feature ranking. One network-host based method involves analyzing keystrokes to determine when malware is attempting a network connection. Registry anomaly detection utilizes machine learning to compare registry changes against a normal baseline to detect malicious changes. Both setups are computationally expensive, but, if improved, could be implemented in a virtual machine set up.

Virtual machines (VM) have been used in a variety of malware detection projects. VMs separated from the host operating system (OS) offer a level of protection from malware trying to subvert anti-virus programs. Also, as VMs are separated from the host OS, a client-server relationship between the two can be established. This eliminates the need for a second computer in solutions requiring a client-server set up. Lightweight OSs running in a virtual machine decrease host system resource requirements, run faster, and are potentially more secure than normal-sized OSs.

Running keystroke analysis and registry anomaly detection programs in a virtual machine protect them from malware and offer additional advantages. The VM can be trimmed down, using the bare minimum of required resources and components to successfully run the detection solution without harming performance. In this research, we propose correlating suspicious network traffic generated by unusual keystrokes with suspicious registry accesses in order to detect malware with a higher degree of accuracy and with a lower false positive rate.

II. EXISTING ALGORITHMS

A. *Detection of kernel-level invariants*

Kernel data structures are often modified by rootkits in an attempt to hide their execution from detection methods. Kernel data structures include control data such as the system call table, jump tables and function pointers. They also include non-control data such as linked lists used for bookkeeping and pseudorandom number generators. Gibraltar [1] automatically generates kernel data structure integrity specifications known as data structure invariants. Invariants are properties that must hold for the lifetime of the data structure. Gibraltar's inference phase creates a baseline of the kernel data structures. During the rootkit detection phase, invariants are compared against the baseline. Any deviation is assumed to indicate the presence of a rootkit. Gibraltar resides on an external computer and captures snapshots of the target system's kernel memory via an external PCI card and reconstructs the kernel data structure. It utilizes Daikon, an invariant inference tool, to infer invariants on the kernel data structures. Gibraltar successfully detected 23 of 23 rootkits during experiments.

Benchmarking utilities determined Gibraltar added a one-half percent runtime monitoring overhead, a very minimal amount. It successfully detected rootkits that modified both control and non-control data structures with an average detection time of twenty seconds. Gibraltar works well in a client-server setting. One server running Gibraltar can manage malware detection on multiple clients.

Gibraltar has downsides as well. It requires a second, *observer* computer to monitor the target computer. Gibraltar cannot be deployed on the target computer. It also has a long startup time. It requires twenty five minutes to take snapshots of kernel memory followed by thirty one minutes to infer invariants. This is a total of fifty six minutes every time the target computer boots up before Gibraltar is ready to monitor it. However, the researchers determined invariant inference can be completed in parallel while the system takes the next snapshot of kernel memory. Gibraltar also infers 236,444 or more invariants. Each of these invariants is very precise. There currently is no way to group the invariants together e.g. broader rules that encompass multiple invariants. The invariants are not portable and are system-specific. Each target must be analyzed every time the system boots. False invariants may be inferred and refined to reduce spurious alerts. Gibraltar also cannot detect transient attacks, that is, rootkits that modify an invariant and revert it back between kernel snapshots.

B. Cloud based malware detection

Cloud anti-virus servers [2] offer enhanced detection of malware. Cloud servers require an analysis engine to scan for malware. Multiple anti-virus programs and detection algorithms can be loaded on the cloud server. A forensics archive serves as a database with which the analysis engine compares malware against. Client computers require an isolated host that interfaces to the cloud server and the system memory or physical disk. An isolated host agent environment allows the host to send requests and provides direct access to host storage. Two prototypes are proposed. The first is based on the Intel Active Management Technology (AMT) combined with the Intel vPro. The second is based on a Virtual Memory Monitor.

Cloud-based anti-virus servers reduce the amount of storage and computational resources required on the client due to the fact no anti-virus resources must be installed on the client. It simplifies management of signature files, as only the information on one computer, the server, must be configured. Also, since servers are typically more powerful than individual workstations, more advanced, sophisticated and computationally expensive heuristics can be employed to determine threat profiles.

Disadvantages include the fact that host agents still require mechanisms to detect and prevent agents that have been disabled or subverted. The first prototype based on Intel AMT uses a blacklist approach. Only 192KB of a blacklist can be stored. This is a very small amount of storage for an ever-increasing amount of malware. Scan frequency is also low. Additionally, attackers can compromise the host operating system or the virtual machine monitor itself, thereby circumventing the detection mechanisms.

C. Joint network-host based malware detection

Joint network-host based malware detection with information theoretic tools [3] detects deviations from a behavioral model baseline derived from a benign data profile. A baseline of keystrokes is determined against which data is compared. This algorithm analyzes perturbations in the distribution of keystrokes used to create network connections. Keystroke entropy increases and session-keystroke mutual information decreases when an endpoint is compromised by self-propagating malware. If both host and network features are correlated, malware detection is increased. The last input from a keyboard or mouse hardware buffer is correlated with every new network session. Only outgoing unicast traffic is analyzed, as firewalls block incoming traffic.

This algorithm attains an almost one hundred percent detection rate with a low false-positive rate. Instead of comparing malware to known signatures, it works based on behavioral analysis. This allows the algorithm to detect previously unknown malware.

Joint network-host based malware detection can be defeated by mimicry attacks. Malware utilizing mimicry attacks hide its traffic in benign traffic. This effectively hides its network traffic from the detection system allowing it to avoid detection. Ill-defined security policies and user privileges pose problems for this detection system. Malware can circumvent the policies and exploit user privileges, allowing it to gain system level privileges and disable the detector.

D. System function call analysis

Rather than employing traditional reverse engineering or debugging techniques, this algorithm extracts malware behavior by observing all system function calls [4]. It controls various parameters of a sandboxed virtual execution environment and analyzes the interaction of malware on the system. It computes similarities and distances between malware behaviors in order to classify malware behaviors. A phylogenetic tree, a type of branching diagram, tracks evolution of malware features and implementations. It shows inferred relationships between entities based upon similarities and differences in characteristics.

This method requires research and analysis work to be performed on known malware before the algorithm can be employed against suspected malware. Malware must first be introduced to the virtual machine sandbox environment for analysis and classification. Once malware has been classified and the phylogenetic tree built, unknown executables can be compared against the tree. Zero-day exploits can be detected based on similar operating characteristics.

E. Feature ranking and machine learning

Computer virus detection can be enhanced via feature ranking and machine learning [5]. This is a combination of the information gain and voted perceptron detection methods. Test and training data are fed into a portable executable (PE) parser. The PE parser extracts windows API calls and converts them into thirty two bit global IDs as features of the training data. Features are then selected based on the information theoretical concept of entropy. The distinguishing power of each feature is then derived by computing its information gain (IG) based on frequencies of appearances in the malicious class and the benign class. A voted perceptron classifier constructs the malware detection classifier. This model was tested with known malware downloaded from an online malware database.

Test results demonstrated a ninety nine percent true positive rate, a ninety nine percent detection rate and a ninety nine percent precision rate. These rates are four to nine percent higher than analysis using either the information gain or voted perceptron respectively.

The algorithm must first be trained and fed test data to build a sample signature database of called APIs. However, once built, it could detect zero-day malware based on similar API calls and behavioral analysis. As signatures are added to the database, the system learns and increases its detection capabilities.

F. Registry anomaly detection

Analyzing registry changes facilitates malware detection [6]. Creating a baseline of normal registry changes allows the algorithm to compare registry changes against that baseline. Anything out of the ordinary, e.g. malicious, triggers an alert. The Registry Anomaly Detector (RAD) requires three components. These components include a Registry Basic Auditing Model (RegBAM), a Model Generator, and an Anomaly Detector.

The RegBAM monitors registry reads and writes. Initially, this data is fed into a database for the model generator. After the baseline registry changes model is created, the RegBAM feeds data into the anomaly detector.

The model generator takes data gathered by the RegBAM and builds a normal usage model. This model represents normal registry usage and can be easily distributed to new machines. This is especially desirable in a large IT enterprise where standard desktop configurations are the norm. Normal registry usage should be similar from computer to computer.

The anomaly detector receives live data from the RegBAM. The detector compares data to the normal usage model and generates a score based on the anomalies in the registry. A user-defined threshold signifies when the anomaly detector should trigger an anomalous event.

One disadvantage is the amount of traffic generated by registry reads and writes. The researchers measured a load of approximately 50,000 registry accesses per hour. The three RAD components can be configured on different machines. The downside to this approach is the increase in network traffic. The tradeoff is network traffic vs. host machine resources.

III. METHODOLOGY

A. Bell-LaPadula model for the host and virtual machine

The foundation of this solution lies in the ability to modify a virtual machine to directly access the host operating system. Virtual machines are currently completely separated from the host OS and have no direct access to its internals. Allowing VMs to directly monitor the host OS is an area of on-going research.

The host and virtual machine shall follow the Bell-LaPadula security model [7]. The virtual machine shall be designated a higher security level than the host it resides on. The host shall follow the simple security property, i.e. the host shall not read up to a higher security level, the VM. The VM shall follow the star property, i.e. the VM shall not write down to a lower security level, the host. We caveat this by explicitly specifying which data the host may write up to the VM. The host shall only feed network packets to the VM for analysis. All other writes to the VM from the host shall be disallowed. Four components, shown in figure 1, comprise the solution. This includes a network traffic monitor, a keystroke analyzer, a registry anomaly detector, and a correlator. The VM shall read keystrokes and registry changes on the host machine. The details of these components will be discussed in Sections III.C - III.E.

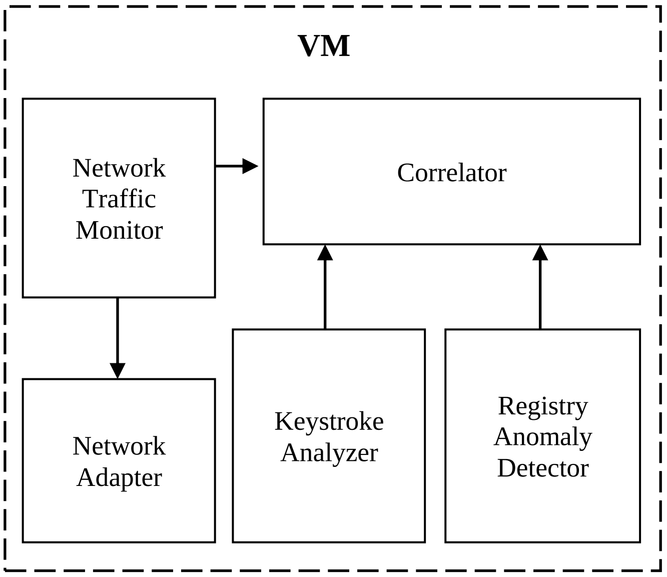

Figure 1. Four components of the malware detection scheme.

Figure 2 below describes data flow between the virtual machine, the host OS, and applications running on the host. Label 1 shows network traffic. The host sends network traffic to the virtual machine for analysis and correlation. This data is then sent back out through the host's network adapter, as the VM contains only a virtual network adapter. The VM does not write any data to the host. Label 2 shows keystroke and registry data flowing to the VM. This data is read from the host by the VM and is not written to the VM by the host.

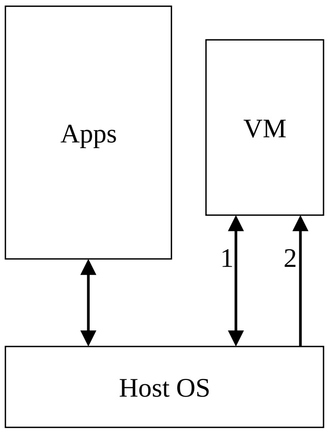

Figure 2. System data flow.

B. Virtual machines to guard malware detection systems

Malware developers usually create their software with stealth in mind. Avoiding detection by antivirus programs, users, and administrators is key. For this reason, malware authors employ a variety of methods to hide their malicious programs. Malware can subvert and disable anti-virus programs and other malware detection methods. It is important to protect anti-malware programs from malware. If malware fails to detect the anti-malware programs, it cannot disable them.

Virtual machines add a level of protection to security solutions. Programs running in a virtual environment are not detectable by anything on the host operating system. The only thing the host OS knows is a virtual machine is running. Any malware that infects the host machine will not be able to attack programs running in the VM.

What does this mean for anti-malware programs? Running anti-malware in a VM prevents any malware that infects the host machine from undermining the anti-malware software. As long as the VM is not infected, malware detection programs will run. Additionally, if VMs directly monitor host internals without installing any software on the host, malware cannot block, terminate, or otherwise disable software the anti-malware solutions depend on. All software resides in the VM.

Secondly, VMs are typically large and resource intensive. Creating a trimmed down, lightweight VM will consume less host processing power and memory. Only the bare minimum of drivers and services needed to run the VM and the four detection components are required. Non-essential elements must be removed. In addition to consuming fewer resources, removing components creates a more secure environment. Fewer components mean less vulnerability.

Multiple lightweight operating systems (including Windows and Linux) that can run in a VM have been created. One example is Damn Small Linux (DSL), a 50mb Linux installation. DSL requires a minimal amount of processor and memory resources. However, DSL contains unnecessary packages, such as Pac Man, that can be removed. Several lightweight Windows installations have been created. nLite allows the user to trim down a Windows installation disk, customizing the installation so only selected components are installed.

Additionally, research has shown trusted virtual machine monitors can boot individual programs into separated, individual virtual machines [8]. These VMs boot directly into the program, without any user interfaces or shells.

C. Correlating keystrokes to network connections in a virtual machine

Of the four components in this solution, correlating keystrokes to network connections requires two of the components. A network traffic analyzer and a keystroke monitor are required. As described in [3], keystrokes are correlated to corresponding network traffic. Their solution uses a joint network-host based approach. We propose feeding network traffic through the virtual machine for analysis and correlation before transmission to the internet.

Virtual machines and their host share a virtual network as shown in Figure 3 below.

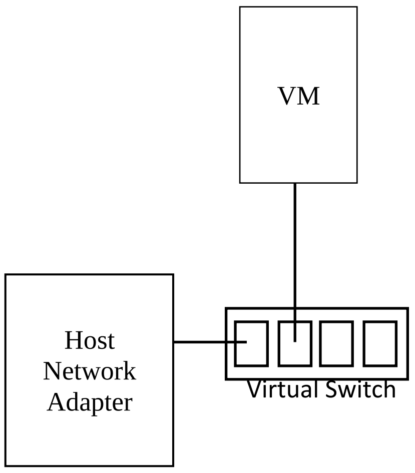

Figure 3. Virtual network between a VM and the host.

Remember in Figure 2 that network traffic flows from the host to the virtual machine. We specifically state that all network traffic must flow through the virtual machine before transmission to the internet. Outbound packets can be forwarded to the VM for analysis by the network traffic monitor. Once routed through the network traffic monitor, the packets are sent to their intended destination.

The second component of this portion is the keystroke analyzer. The keystroke analyzer resides on the VM and requires direct access to the host. It reads down to the host to monitor keystrokes. Each keystroke shall be logged and stored for correlation to a network packet. The solution described in [3] correlates the keystrokes and packets through the use of timestamps. Timestamps are more important when the monitors reside in the VM. The generated packets will retain the same timestamp. Additional delay between the VM and host may cause keystroke timestamps to be slightly later than the actual time. Careful testing of timing is necessary to determine timing delays introduced by the components being inside a virtual machine. It is possible that the additional time for the network packets to arrive at the network traffic monitor could result in it being correlated to the wrong keystroke.

D. Monitoring the host registry from a virtual machine

The third component of this solution is the registry anomaly detector. The authors of [6] propose storing the system behavior model in the registry. This allows the RAD to monitor the baseline model, securing it from malicious changes. The training data gathered for the model comprised 500,000 records, which, when added to the registry, would greatly increase the size. Moving the RAD to a virtual machine would keep the host registry at a normal size, while retaining the desired security.

The main requirement is to directly access the host OS's registry. The RAD proposed in [6] allows the components to be split among systems, with, at a minimum, the RegBAM remaining on the computer being monitored. We propose putting all components in the virtual machine and allowing the RegBAM direct access to the host's registry.

Additionally, the RegBAM needs to be modified to include timestamps. Each registry read or write requires an associated timestamp. The RAD works in real time to detect registry changes, but all changes require a timestamp to allow correlation with suspicious network traffic.

E. Putting it all together

We propose correlating suspicious network traffic with suspicious registry accesses. The probability of detecting malicious software will increase while simultaneously lowering false positive rates through correlation of potentially malicious traffic and potentially malicious registry accesses. If a suspicious network connection is made following an unusual keystroke corresponds to a recent abnormal registry access, the likelihood of malware activity increases. Both components showed high success rates of detection. Correlating both to each other will further increase detection rates and confidence. A lower confidence registry access when correlated to a suspicious network connection may signify the presence of malware that would otherwise fall below detection thresholds.

The RAD, keystroke analyzer, and network traffic monitor looks for specific portions or products of the host. The final component of the virtual machine is the correlator. The correlator works in two parts. As shown in Figure 4, the algorithm consist of two main parts in VM. The first part correlates keystrokes to network traffic. The second part correlates results of part one with output from the RAD.

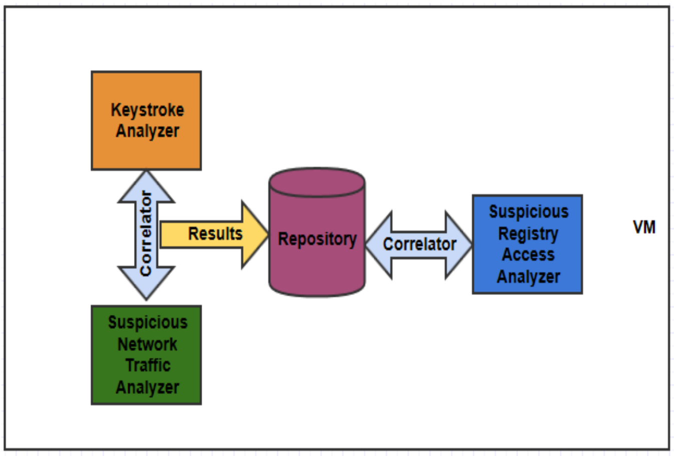
Keystroke Analyzer
Correlator
Results
Repository
Correlator
Suspicious Registry Access Analyzer
VM
Suspicious Network Traffic Analyzer

Figure 4. High Level Design Diagram

The authors of [3] already correlate keystrokes with network traffic. Due to the fact the components are in a VM, the algorithm will most likely need to be modified to account for timing delays as information is transferred to the virtual machine. Timestamps from suspicious registry accesses will be correlated to network traffic and keystrokes. The correlator can be triggered by either a suspicious RAD report or a suspicious network traffic report. Once triggered, it polls the other for recent activity with a similar timestamp. Reports are analyzed and a confidence assigned based on how malicious the activity appears.

IV. CONCLUSION

Recent advances in non-signature-based malware detection have proven effective in research and testing. We have shown how virtual machines can be used to provide a secure environment for anti-malware solutions, helping to protect them from malware that attempts to disable or otherwise harm detection methods. We expand upon the work of [3], [6], and [7] to correlate suspicious network traffic generated by unusual keystroke patterns with suspicious registry accesses. By correlating these together, we theorize a resulting higher detection rate with a lower amount of false positives.

V. FUTURE WORK

We plan further research to support and test our hypotheses. A key component of future research is to create a connection between the host operating system and the virtual machine. This connection needs to act as a diode, allowing the virtual machine to monitor the host's registry and keystrokes, but disallowing all interaction with the VM initiated by the host. We need to trim down a virtual machine to determine the best balance between host performance and algorithm speeds. The more we trim the virtual machine and its operating system, the more efficient the host should run, but the longer it may take our solution to process.

We also plan to gather data regarding actions of malware. We intend to find out the percentage of malware that generates network traffic and the percentage of malware that modifies the registry. This information will allow us to calculate the overall improvement in the ability to detect malware by correlating network traffic with registry accesses.

Timestamps and network delay need additional research. By running our solution in a virtual machine, we'd like to find the answer to see if there are any timing delays introduced that may cause the wrong keystrokes to be correlated to network packets? We also intend to determine timing correlation between malicious registry changes and start of network traffic flow.

REFERENCES

[1] A. Baliga, V. Ganapathy, and L. Iftode, "Detecting kernel-level rootkits using data structure invariants," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 670-685, Sept-Oct, 2011.

[2] C. Rozas, H. Khosravi, D. K. Sunder and Y. Bulygin, "Enhanced detection of malware," *Intel Tech. Jour.*, vol. 13, no. 2, pp. 6-15, Jun, 2009.

[3] S. Khayam, A. Ashfaq and H. Rahda, "Joint network-host based malware detection using information-theoretic tools," *Jour. Compute. Virology*, vol. 7, no. 2, pp. 159-172, May, 2011.

[4] G. Wagener, R. State and A. Dulaunoy, "Malware behaviour analysis," *Jour. Compute. Virology*, vol. 4, no. 4, pp. 279-287, Nov, 2008.

[5] A. Altaher, S. Ramadass and A. Ali, "Computer virus detection using features ranking and machine learning," *Australian Jour. Basic & Applied Sciences*, vol. 5, no. 9, pp. 1482-1486, 2011.

[6] F. Apap, A. Honig, S. Hershkop, E. Eskin and S. Stolfo, "Detecting malicious software by monitoring anomalous windows registry accesses, *5th International Symposium on Recent Advances in Intrusion Detection*, Zurich, Switzerland, 2002.

[7] D.E. Elliot and L. J. LaPadula, "Secure computer systems: a mathematical model," MITRE Corp., Bedford, MA, Tech. Rep. 2547, May 31, 1973.

[8] T. Garfinkle, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, "Terra: a virtual machine-based platform for trusted computing," *19th Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003, pp. 193-206.