West Virginia University
Lane Department of Computer Science and Electrical Engineering
Technical Report

MALWARE DETECTION ON GENERAL-PURPOSE COMPUTERS USING POWER CONSUMPTION MONITORING: A PROOF OF CONCEPT AND CASE STUDY

April 28, 2017

Malware Detection on General-Purpose Computers Using Power Consumption Monitoring: A Proof of Concept and Case Study

Jarilyn M. Hernández Jiménez*†, Jeffrey A. Nichols*, Katerina Goseva-Popstojanova†,
Stacy Prowell*, and Robert A. Bridges*
* Computational Science and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831
{hernandezjm1, nicholsja2, prowellsj, bridgesra}@ornl.gov
†Lane Department of Computer Science and Electrical Engineering, West Virginia University, WV, Morgantown, 26506
{jhernan7, katerina.goseva}@mail.wvu.edu

***Abstract*—Malware detection is challenging when faced with automatically generated and polymorphic malware, as well as with rootkits, which are exceptionally hard to detect. In an attempt to contribute towards addressing these challenges, we conducted a proof of concept study that explored the use of power consumption for detection of malware presence in a general-purpose computer. The results of our experiments indicate that malware indeed leaves a signal on the power consumption of a general-purpose computer. Specifically, for the case study based on two different rootkits, the data collected at the +12V rails on the motherboard showed the most noticeable increment of the power consumption after the computer was infected. Our future work includes experimenting with more malware examples and workloads, and developing data analytics approach for automatic malware detection based on power consumption.**

I. Introduction

Polymorphic malware can bypass signature-based detection methods and simple heuristic detection techniques by slightly changing the instructions of an existing malware sample. These new malware instances are called variants. Although these variants appear to be different programs from the viewpoint of signature-based anti-virus scanners, they exhibit similar functionality to their predecessor. Consequently, these new malware variants can bypass traditional detection methods until a signature for them can be identified and incorporated into detection software [1].

Authors of malware detection systems have attempted to address this problem by using other methods that are more powerful than signature matching; for example, byte frequency [2], general similarity measures [3], and behavioral analysis [4] are among the proposed techniques. A common weakness of these detection methods is that they are executed on the same machine they are monitoring. Hence, successful attackers could disable the monitoring software or modify it to prevent detection after gaining entry to the system [5]. This behavior is evidenced by rootkits, a particularly insidious subclass of malware. Rootkits are a type of computer malware that were created to hide themselves and elude intrusion detection systems once they gain unauthorized access to a computer system [6].

Previous work has also explored the idea of detecting the presence of malware by monitoring the power consumption of mobile devices, embedded systems, and software define radio. However, to the best of our knowledge, no one has explored if malware can be detected by monitoring the power consumption on general-purpose computers.

Our goal in this paper is to prove the hypothesis that in order to mask themselves, rootkits will require a detectable change in the power consumption. Particularly, we are addressing the following research question: can we detect rootkits on general-purpose computers by analyzing only the power consumption? To this end we built a testbed and designed an experimental setup in which the power consumption was recorded for a sequence of events running on a Windows operating system. This work focuses only on rootkits because they are commonly associated with the establishment of advanced persistent threats and pose serious danger to our nation's computer systems. Preliminary results showed that malware indeed leaves a signal on the power consumption of a general-purpose computer. Specifically, monitoring the +12V rails on the motherboard was the most useful for identifying the increase in the power consumption after the general-purpose computer was infected by malware.

The paper proceeds with related work in Section II, followed by the experimental design in Section III, which includes the hardware and software setups used for collecting the power data, the experimental machine's execution of tasks, and descriptions of the rootkit. Section IV presents the results of the feasibility study. Finally, conclusion and promising directions for future research are discussed in Section V.

II. Related Work

Several works have used power consumption metrics for malware detection purposes. These methods have been tested

on mobile devices [7], [8], embedded systems [9], and software defined radio [10], [11], [12].

The work by Hoffman et al. [7] explored if malware can be detected on smartphones by analyzing their power consumption. This method failed due to the noise caused by unpredictable factors, such as user interaction and the mobile's signal strength. On the other side, the approach presented by Yang et al. [8] demonstrated that malware can be detected by monitoring the power consumption of smartphones. The difference between these two works is mainly in the type of smartphones used in the experiments. First method [7] focused mainly on "old" devices (HTC-Nexus One and Samsung Galaxy Nexus), while the second method [8] focused on modern devices (Samsung Galaxy S5 and LG G2). Although PowerTutor [13] was used for the data collection in both works, this tool may have been updated between the time these two experiments were conducted, influencing the precision of the collected data and skewing the results.

Another method that monitors the power consumption on embedded systems with the objective of detecting malware was presented by Clark et al. [9]. Supervised machine learning techniques, such as 3-Nearest Neighbor, Multilayer Perceptron, and Random Forest, were used to analyze alternating current (AC) and to detect discrepancies among the power profiles. Even though the proposed approach share several similarities with this work, the main difference is that the work in [9] focused on monitoring the AC outlet, while we are monitoring several direct current (DC) channels. The problem with AC is that the current changes direction periodically, and because the current changes its direction the voltage reverses making the analog circuits much susceptible to noise.

Similarly, power-based malware detection for software defined radio was explored by González et al. [10], [14]. This approach relied on extracting distinctive power consumption signatures and used pattern recognition techniques to determine if they matched the expected behaviors. This research was expanded, and used by the PFP firm (http://pfpcyber.com), which developed a commercial product that detect anomalies on a device by analyzing its power consumption. This approach can also be applicable to embedded devices [11], [12]. The main difference between this approach and our work is that we monitor all the rails attached to the motherboard plus the CPU, while PFP is monitoring the power consumption of the device by placing a sensor on the processor's board as close to the power pins as possible.

It appears that there are no published research works focused on testing the use of power consumption monitoring in support of malware detection in general, with respect to the detection of rootkits in particular.

III. Experimental Design and Data Collection

The objective of our experiments is to analyze the power consumption of a general-purpose computer in order to detect the presence of rootkits. Our work is based on the hypothesis that rootkits can be detected by the anomalies they cause in the DC power consumption of the general-purpose computer. Specifically, we are interested in determining if there is a difference in the power profiles between the normal and anomalous behavior (i.e., after infection).

A. Hardware Configuration

Our experimental system is a Dell OptiPlex 755 with a clean installation of 32-bit Windows 7. The instrumentation for our experiments was a Data Acquisition system (DAQ), Model Number: USB-1608G Series [15]. The DAQ connects to the device's motherboard power connector, and the voltage and current are collected on each of the DC power channels. The communication between this machine and the experimental machine was established through USB port. The DAQ provides relatively high-resolution power data, is able to sample at a rate of 250KHz, and can monitor up to 16 channels. Besides the DAQ, we also used an eight inch ATX power extender cable that had one male and one female 24-pin connector. The 24-pin male connector was attached to the motherboard, and the 24-pin female connector was attached to the power supply (PSU).

Each group of wires on the PSU were connected to a single overcurrent protection (OCP) circuit that is called a *rail*. A PSU has three voltage rails: *+3.3V*, *+5V*, and *+12V*. Table I provides a list of the devices that are typically powered by these voltage rails. The +3.3V rails or the +5V rails are typically used by the digital electronic components and circuits in the system [16]. Some examples of these components are adapter cards and disk drive logic boards. On the other hand, the disk drive motors and the fans use the +12V rails [16]. Besides disk drive motors and newer CPU voltage regulators, the +12V supply is used by any cooling fans in the system [16].

TABLE I: Voltage rail usage for a general-purpose computer

Rail	Devices Powered
+3.3V	chipsets, some DIMMs, PCI/AGP/PCIe cards, miscellaneous chips
+5V	disk drive logic, low-voltage motors, SIMMs, PCI/AGP/ISA cards, voltage regulators
+12V	motors, high-output voltage regulators, AGP/PCIe cards
+12V CPU	CPU

Acronyms:

- SIMM = Single Inline Memory Module
- DIMM = Dual Inline Memory Module
- PCI = Peripheral Component Interconnect
- PCIe = PCI Express
- AGP = Accelerated Graphics Port
- ISA = Industry Standard Architecture
- CPU = Central Processing Unit

To ensure the power data was collected adequately, we tested three hardware configurations. The first hardware configuration monitored a total of eleven DC power channels (four pins had a signal of +3.3V, five pins had a signal of +5V, and two pins had a signal of +12V). When using this configuration, the voltage levels were obtained for each of the channels that

were monitored. However, since we were interested in power, both the voltage and current were required. To address this challenge, a DC voltage and current sense PCB was used.

The DC voltage and current sense PCB determines the DC current by measuring the voltage drop across a shunt resistor, and then converts that current to analog voltage output [17]. The PCBs were soldered to those wires on the ATX power extender cable that we were interested in monitoring.

Leaving a total of thirteen DC power channels to be monitored (ten of them were used to measure the current and the other three channels were used to measure the voltage). Since the value of the voltage was the same, we measured the voltage as a group, that is, one voltage value for all the rails that were +3.3V, one value for all the rails that were +5V, and one value for all the rails that were +12V. While testing this hardware configuration, we noticed that there were two +12V rails that were powering the CPU of the experimental machine. Particularly, these +12V rails were separate from the rails that we were already monitoring on the ATX power extender cable. These +12V rails were connected from the PSU to a 4-pin ATX12V power connector on the motherboard. Including these rails, we ended up monitoring a total of fifteen channels. [1]

Monitoring fifteen channels at the same time was challenging because, when post-processing, we had to sum several measured currents together. To simplify the hardware configuration, we evaluated other options that could help us to reduce the number of channels to be monitored. After some exploring we found that all wires from the same voltage value were soldered together on the same contact point on the power supply. This means that all the +3.3V rails were connected to the same contact point, and the same was true for the +5V rails, and the +12V rails. Figure 1(left) shows the voltage and current sense PCB that was used for the second hardware configuration, while Figure 1(right) shows how the +12V rails were soldered together on the same contact point on the PSU.

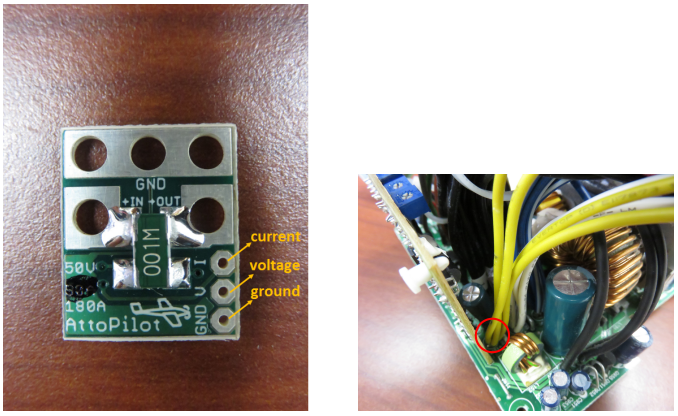

Fig. 1: (Left) Voltage and current sensor PCB used on the experiments for the second hardware configuration (Right) +12V rails soldered on the same contact point on the PSU.

[1]A survey of other machines was made to verify that general-purpose computers have this 4-pin ATX12V power connector. More than 20 computers were verified and all of them had the 4-pin ATX12V power connector.

The third hardware configuration emerged from this observation. We grouped all the +3.3V rails on the same voltage and current sense PCB which was attached to the ATX power extender cable; the same was done for the +5V rails and the +12V rails. Figure 2(left) shows the third hardware configuration, while Figure 2(right) shows how the wires from the ATX power extender cable were hooked to the DAQ. As can be seen from Figure 2(right), for each channel on the DAQ we hooked a wire for the current (black wire), the voltage (red wire), and the ground (silver wire).

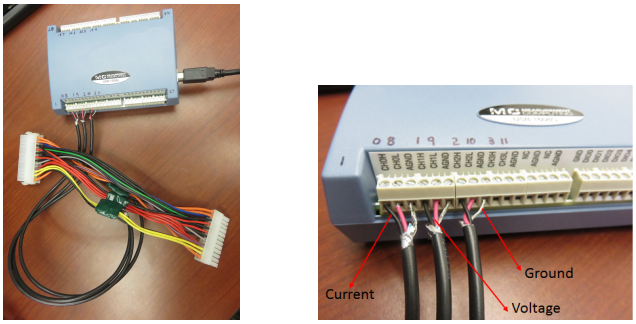

Fig. 2: (Left) Third hardware configuration used during the experiments (Right) Wires attached to DAQ.

Grouping the rails reduced the numbers of channels to be monitored to six—three channels for measuring the current, and the other three channels for measuring the voltage. In addition, we also included the two +12V rails that power the CPU. Overall, instead of monitoring fifteen channels, we reduced the number to eight: 4 voltage channels and 4 corresponding current channels. This configuration was the one used for the experiments and data collection described here.

B. Software Configuration

Initially, we used a tool called *TracerDAQ Pro* (version 2.3.1.0), which is an out-of-the box virtual instrument that acquires and displays data [15]. This tool ran on a different machine (data repository) in order to provide integrity during the experimentation process. The acquired data from the experimental machine was stored as a CSV file on the data repository.

As our experimental design evolved, we found that TracerDAQ Pro was not suitable for obtaining precise power data. To address this issue we developed our own Visual Basic program. There were three advantages to using our own software versus using the supplied software: (1) data has 16 bits of precision; (2) we have control over the sample recording rates and sample timing (3) we were able to make additional real-time calculations that helped us to verify the obtained power data.

Another application used in our experiments is called *Clonezilla* [18]. Clonezilla is a partition and disk imaging/cloning program. This tool was used to ensure we had a consistent, clean installation of Windows. We used Clonezilla to create an exact copy of the master hard drive and this hard drive was not exposed to malware.

C. Data Collection

The power consumption of the general-purpose computer was collected in two different scenarios: normal behavior (no rootkit running on the system) and anomalous behavior (a rootkit was running on the system). For the data collection workflow, we assumed a clean installation of Windows, then power data was collected and labeled as normal. Subsequently, the experimental machine was infected and power data was collected and labeled as anomalous. For this case study we infected the general-purpose computer with two rootkits: Alureon and Pihar.

A segregated network was created to ensure the malware will not spread around the main network. The segregated network consisted of an experimental machine, a data collection repository, a hub, and a cellular data connection. The data collection repository connects to the personal hotspot, and then through the hub we shared the wireless connection with the experimental machine. Two advantages from the use of segregated network are: (1) allowing rootkits to behave normally, while avoiding the possibility of infecting other machines on the network; and (2) allowing us to monitor, record, and analyze the experimental machine's network traffic.

Wireshark was used to collect the network traffic of the experimental machine and to validate that the experimental machine was successfully infected with the rootkits being tested. As part of the network traffic analysis, we organized the protocols on the PCAP file by alphabetical order and then focused only on the column for the Domain Name System (DNS) protocol. From the domains that were captured, one of them got our attention (*term0l5ter12.com*). Several websites [19], [20] had this domain registered as malicious. After all these analyses, we were certain that the experimental machine was successfully infected with both rootkits.

To initialize the data collection process, we wrote two scripts: a Python script that executes a sequence of events, and a C++ program that inserted what we called a *marker*. The objective of the Python script was to ensure repeatability, while the objective of the marker was to insert a signal into the measured power data to mark the start and end points for each of the events. IE was chosen because the Alureon and Pihar rootkits affect the performance of browsers [21], [22].

When the Python script is executed, it launches two markers before the experimental machine goes idle for a minute. Then the Python script opens ten windows of IE each with 5 seconds delay. Figure 3 shows data collected after the Python script was executed for the +12V CPU rail prior and after the infection with the Alureon rootkit. These events (idle, opening IE, booting/rebooting) were recorded during three states: (1) prior to infection, (2) after infection, and (3) after infection plus reboot. In order to segment these sections of the power profile, we used the marker to stress the CPU of the experimental machine for five seconds. The Python script places markers in the power data before and after the events were recorded. The advantage of using these markers is that they allow us to understand when a particular event occurs and how long it takes to complete its execution. This workflow was completed three times for the four rails that were monitored.

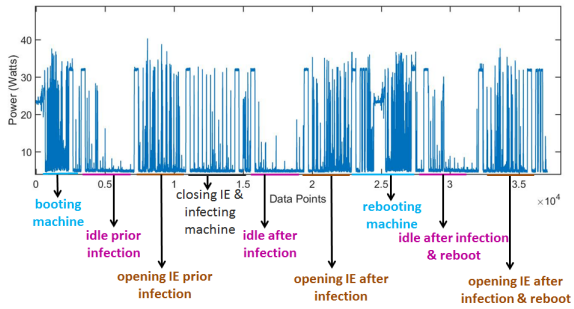

Fig. 3: Sequence of events after the Python script was executed

The first rootkit, Alureon, also known as *TDL4* or *TDSS*, is a Trojan that allows an attacker to intercept incoming and outgoing Internet traffic in order to gather confidential information such as user names, passwords, and credit card data [23]. There are several generations of this type of malware, and for our experiments, we used the fourth generation [24]. Typically, it infects a computer via drive-by download through a questionable website, often a distributor of pornography or pirated media [25]. Once Alureon is installed on the machine, the software searches the system for any competitor's malware and removes it. It also uses an encryption algorithm to hide its communications from traffic analysis tools that are sometimes used to detect suspicious transmissions [25]. Furthermore, this rootkit can manipulate the master boot record (MBR) of the computer to ensure that it is loaded early during the bootup process so that it can interfere with the loading of the operating system [26]. The second rootkit, which is a variant of Alureon, is a Trojan called *Purple Haze* (also known as *Pihar*). Like Alureon, this rootkit can modify the MBR of the machine, as well as changing system settings and reconfiguring the Windows registry. Its rootkit capabilities include disabling the antivirus software to keep itself hidden [27].

D. Data Pre-processing

As part of the data pre-processing, the voltage and current for the monitored rails were multiplied to obtain the power consumption of the general-purpose computer. To plot and interpret the power data, we used MATLAB. After obtaining the power data, the next step was to separate the events based on the start and end point.

To obtain the indexes we wrote a MATLAB script that returns the start and end point of all the markers that appeared on the dataset. For this case study, there are a total of eighteen markers. Once we had the start and end point for each event, the next step was to compare those events that were related to each other. Specifically, we were interested in the following comparisons: (1) when the machine was booting prior to infection versus when the machine was rebooting after infection; (2) idle prior to infection versus idle after infection; (3) idle prior to infection versus idle after infection

and reboot; (4) when opening IE windows prior to infection versus when opening IE windows after infection (5) when opening IE windows prior to infection versus when opening IE windows after infection and reboot.

IV. Data Analysis and Case Study Results

The primary goal of this proof of concept is to determine if there is a difference in the power consumption of a general-purpose computer after malware infection. To prove or disprove this hypothesis, several experiments were conducted and power profiles were collected for specific events (idle, opening IE, and booting/rebooting). This was done for the rootkits Alureon and Pihar. For each rootkit there were three datasets. Each dataset contains the power consumption obtained for each of the rails that were monitored. In other words, each dataset contains the power profiles of all the sequence of events that were recorded for each one of the rails. The comparison between the normal and anomalous state was done for each of the events that were recorded on the four rails. Five graphs were generated for each monitored rail. The x axis for each of these graphs shows "Data Points", which refers to the total of power readings that were sampled every 10 milliseconds. For example, if a graph shows 3,000 data points that would be equivalent to 30 seconds.

A. +3.3V Rails

These rails are typically used by digital electronic components and circuits in the system, such as memory. When comparing the power profiles of booting prior to infection versus when it was rebooting after infection, we noticed that at the beginning the power consumption was lower and subsequently both events kept their power consumption similar to each other. Regarding the other events (idle and opening IE), results showed that the difference in the power consumption cannot be established by the naked eye. After analyzing all six datasets (three datasets per rootkit), we concluded that the +3.3V rails are not very useful for detecting different behaviors between the normal and anomalous power profiles because these rails are used to power up memory, and that component does not consume as much power as the hard drive or CPU.

B. +5V Rails

For all datasets, when comparing booting prior to infection with the rebooting after infection for +5V rails , we noticed the same behavior as the +3.3V rails, that is the power consumption after infection was lower at the beginning of the initialization process, but later it kept the same pace as the normal behavior. Hence, comparing booting prior to infection versus booting after infection for the +5V rails is not sufficient to distinguish between normal and anomalous behavior.

When we compared idle prior to infection versus idle after infection with Alureon we obtained an increment in the power consumption after the general-purpose computer was infected for two out of the three datasets (66.67% of the time), while for Pihar we noticed an increment in the power consumption for all datasets (100% of the time). However, when comparing idle prior to infection versus idle after infection and reboot for both rootkits, we noticed that the power profiles for both scenarios (normal and anomalous) were at the same level. In other words, a distinguishable difference cannot be made by the naked eye. Furthermore, when comparing all the graphs in which the general-purpose computer was idle we noticed a delay in the power data after the general-purpose computer was infected. We believe this delay is because after the infection more processes are running and this extra work consumes more power. Figure 4 shows the power consumption after infecting the general-purpose computer with the Alureon rootkit. As can be seen from Figure 4, the power consumption in the idle state was higher after the infection than prior to infection. Hence, this comparison is a good criterion for detecting malware through the power consumption.

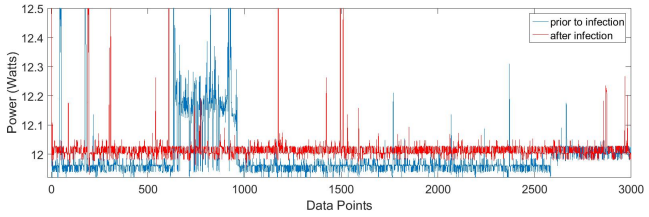

Fig. 4: Power consumption for idle prior to infection vs. idle after infection with Alureon for the +5V rails

When IE was opened prior to infection versus after the infection with Alureon, we noticed an increment in the power consumption after infection for two out of three datasets (66.67% of the time). In the case of the Pihar rootkit, this behavior was seen only in one out of three datasets (33.33% of the time). Figure 5 shows the power consumption after opening IE prior to infection versus after infection. From Figure 5 we can see an increment in the power consumption when some IE windows were opened. Interestingly, this increment was seen when some windows of IE were jammed. This was consistent with the behavior we saw during the data collection process and later was confirmed when analyzing the PCAP file. Based on network traffic collected by Wireshark, we noticed that Alureon was trying to redirect the search engine to advertisement websites. However, when IE was opened prior to infection versus after the infection and reboot for both rootkits, a difference by the naked eye could not be established.

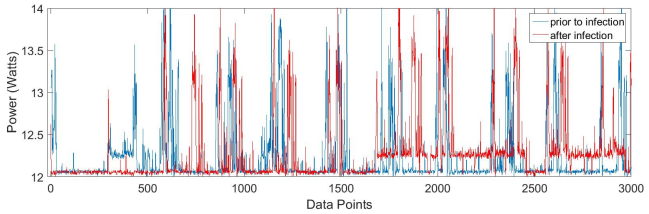

Fig. 5: Power consumption for opening IE prior to infection vs. opening IE after infection with Alureon for the +5V rails

C. +12V Rails on the Motherboard

The +12V rails on the motherboard are used to power up the disk drive motors and the fans. For one of the Alureon datasets results showed that the power consumption was higher after the infection compared to when it was booted prior to infection (33.33% of the time). However for the other two datasets, we saw similar behavior as in the case of +3.3V and +5V rails. Figure 6 shows an increment in the power consumption after the general-purpose computer was infected during the initialization process. In the case of Pihar, an increment in the power consumption was noticeable on two out of three datasets (66.67% of the time).

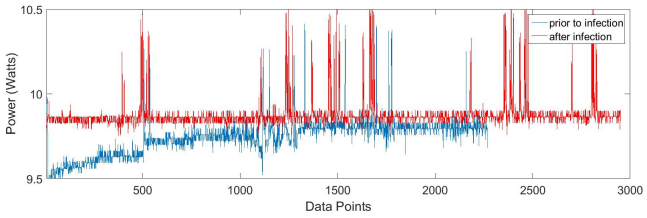

Fig. 6: Power consumption for booting prior to infection vs. booting after infection with Alureon for the +12V rails on the motherboard

When comparing the idle state (idle prior to infection versus idle after infection and idle prior to infection versus idle after infection and reboot), results for Alureon showed an increment in the power consumption after infection for two out of the three datasets (66.67% of the time). Similar increment was seen in all three datasets of Pihar (100% of the time). Figure 7 shows an increment in the power consumption when comparing idle prior to infection versus idle after infection and reboot for the Alureon rootkit.

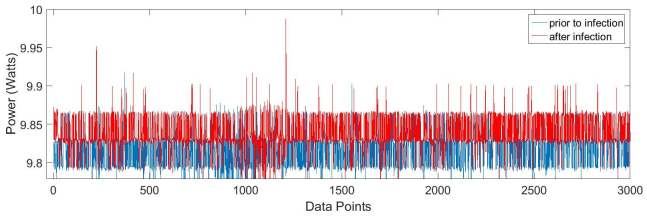

Fig. 7: Power consumption for idle prior to infection vs. idle after infection and reboot with Alureon for the +12V rails on the motherboard

Nonetheless, when comparing IE (IE prior to infection versus after infection and IE prior to infection versus after infection and reboot), results for Alureon showed that an increment in the power consumption after infection can be seen in only one of the datasets (33.33% of the time). Figure 8 shows an increment in the power consumption when comparing IE prior to infection versus IE after the Alureon infection and reboot. After analyzing the +12V rails on the motherboard, we concluded these rails are very useful when analyzing the normal and anomalous power profiles.

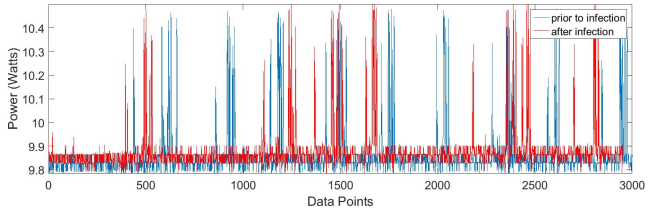

Fig. 8: Power consumption for opening IE prior to infection vs. opening IE after infection and reboot with Alureon for the +12V rails on the motherboard

When comparing IE prior to infection versus IE after infection for Pihar, we noticed an increment in the power consumption after infection for one out of the three datasets (33.33% of the time). Interestingly, when comparing IE prior to infection versus IE after infection and reboot we noticed the power consumption of the general-purpose computer was higher after infection for all datasets (100% of the time).

D. +12V CPU Rails

The +12V CPU rails are separate from the +12V rails on the motherboard (monitored in the PSU). They are used to power the CPU or GPU of a general-purpose computer. The +12V rails on the motherboard are used to power disk drive motors and fans

The comparison between the power consumption when the general-purpose computer was booting prior to infection versus when it was booting after infection showed that at the beginning of the initialization process the power consumption was higher prior to infection for both rootkits. However, at some point during the initialization, an increment in the power consumption after infection was noticeable. This comparison by itself does not provide information that can help us to distinguish between normal and anomalous behavior because of the presence of noise. Noise is expected during the booting and rebooting process because the system is executing several processes simultaneously, so even if the malware is present, its challenging to differentiate between normal and anomalous states.

In the case of idle (idle prior to infection versus after infection and idle prior to infection versus after infection and reboot), we noticed that the power consumption for both rootkits in the normal and anomalous scenarios were similar. However, there were some higher spikes after infection. We believe these spikes were generated when the system was executing normal “non malicious processes”. Similarly, these spikes were also seen in the +5V rails. To be sure about the cause of these spikes, as part of our future work, we plan to collect other parameters such as kernel events, registry files, or syslogs of the general-purpose computer and correlate this information with the power consumption.

A similar behavior was noticeable during IE execution (IE prior to infection versus after infection and IE prior to infection versus after infection and reboot). Results showed that for both normal and anomalous power profiles, the power consumption

was similar. In addition, some delays were seen on the general-purpose computer after it was infected. Figure 9 showsthe power consumption for opening IE prior to infection versus opening IE after infection with Alureon

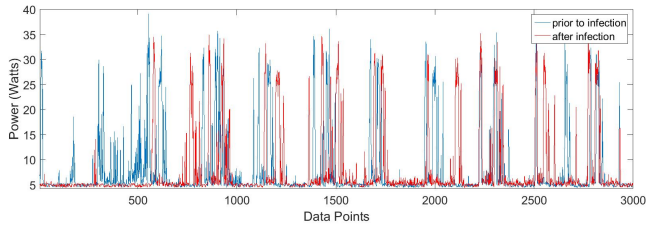

Fig. 9: Power consumption for opening IE prior to infection vs. opening IE after infection with Alureon for the +12V CPU rails

After analyzing all six datasets (three datasets per rootkit), we concluded that a distinguishable difference cannot be made by the naked eye when analyzing the normal and anomalous power profiles for the +12V CPU rails. These results are not the ones we expected because by monitoring the CPU of the general-purpose computer we thought these rails would be more informative. However, we are aware that many processes are running and this extra work consumes more power making it difficult to establish a difference by the naked eye. However, it is possible that the normal and anomalous power profiles may be distinguished by using machine learning algorithms.

V. Conclusions

In this paper we presented a proof of concept whose objective was to investigate whether malware leaves a signal on the power consumption of the general-purpose computer. Power data was collected for four rails (+3.3V, +5V, +12V, and +12V CPU) in two different states (normal and anomalous) for two different rootkits. A comparison between the power consumption of the normal and anomalous state was made for each of the events that were recorded.

The results showed that malware undoubtedly leaves a detectable signal on the power consumption of a general-purpose computer. The signal on the +12V rails on the motherboard was the most useful when identifying an increment in the power consumption after the machine was infected. Results for Alureon showed that when the general-purpose computer was idle (idle prior to infection versus idle after infection and idle prior to infection versus idle after infection and reboot) in a 66.67% of the time an increment in the power profiles was noticeable by the naked eye, while for Pihar this increment in power was seen in 100% of the time. For both Alureon and Pihar, there was a 33.33% of the time in which a notable power signal was seen after the Alureon infection when IE was opened (IE prior to infection versus IE after infection and IE prior to infection versus IE after infection and reboot). In the case of Pihar, 33.33% of the time an increment was noticeable in the power after infection when opening IE prior to infection versus after infection. When comparing IE prior to infection versus infection and reboot with Pihar, we noticed an increment in the power consumption 100% of the time.

Besides the +12V rails, the +5V rails are also a valuable parameter to obtain an increment in the power consumption after infection. Results for Alureon showed that 66.67% of the time there was an increment in the power consumption when comparing idle prior to infection versus idle after infection and when comparing IE prior to infection versus after infection. In the case of Pihar, a noticeable increment in the power consumption was seen 100% of the time when comparing idle prior to infection versus idle after infection. However, when comparing opening IE prior to infection versus after infection with Pihar we noticed an increment in the power consumption after infection only 33.33% of the time.

While we obtained promising results, more rootkit samples and complex data analytics are needed to test and validate this approach. In addition, while all the processes running on the machine consumes power, distinguishing between the normal and anomalous behavior for the general-purpose computer is a challenge because this device is not limited to a certain amount of instructions. Increasing in this way the false positives.

As part of our future work we intend to include more rootkit samples and workloads in the experimental design and data collection process. Furthermore, we plan to propose an approach that can minimize the number for false positives. In addition, we plan to incorporate machine learning techniques to automatically distinguish between the normal and anomalous power profiles and detect malware.

Acknowledgments

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy. This material is based upon work supported by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Building Technologies Office. Katerina Goseva-Popstojanova's work was funded in part by the NSF under grant CNS-1618629. The authors thank Darren Lopossser from the Research Instrumentation group at ORNL for his contributions to this project by providing electronics and sensor support.

References

[1] P. O'Kane, S. Sezer, and K. McLaughlin, "Obfuscation: The hidden malware," Security & Privacy, IEEE, vol. 9, no. 5, pp. 41–47, 2011.

[2] S. Yu, S. Zhou, and R. Yang, "Detecting malware variants by byte frequency," Journal of Networks, vol. 6, no. 4, pp. 638–645, Apr. 2011.

[3] P. Vinod, P. Rakesh, and G. Alphy, "Similarity measure for obfuscated malware analysis," Information Security in Diverse Computing Environments, p. 180, 2014.

[4] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," in NSDI, 2010, pp. 391–404.

[5] S. Sutherland, "10 evil user tricks for bypassing anti-virus," Retrieved from: https://blog.netspi.com/10-evil-user-tricks-for-bypassing-anti-virus/, 2013, accessed: 2015-03-12.

[6] Tech-FAQ, "Rootkit," Retrieved from: http://www.tech-faq.com/rootkit.html, 2016, accessed: 2015-03-12.

[7] J. Hoffmann, S. Neumann, and T. Holz, "Mobile malware detection based on energy fingerprints—a dead end?" in Research in Attacks, Intrusions, and Defenses. Springer, 2013, pp. 348–368.

[8] H. Yang and R. Tang, "Power consumption based android malware detection," Journal of Electrical and Computer Engineering, vol. 2016, 2016.

[9] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb et al., "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices," in HealthTech, 2013.

[10] C. R. A. Gonzalez and J. H. Reed, "Power fingerprinting in SDR & CR integrity assessment," in Military Communications Conference, 2009. MILCOM 2009. IEEE. IEEE, 2009, pp. 1–7.

[11] C. A. Gonzalez and A. Hinton, "Detecting malicious software execution in programmable logic controllers using power fingerprinting," in International Conference on Critical Infrastructure Protection. Springer, 2014, pp. 15–27.

[12] J. H. Reed and C. R. A. Gonzalez, "Enhancing smart grid cyber security using power fingerprinting: Integrity assessment and intrusion detection," in Future of Instrumentation International Workshop (FIIW), 2012. IEEE, 2012, pp. 1–3.

[13] "Powertutor: A power monitor for android-based mobile platforms," Retrieved from: http://ziyang.eecs.umich.edu/projects/powertutor/, 2017, accessed: 2017-04-06.

[14] C. R. A. González and J. H. Reed, "Power fingerprinting in SDR integrity assessment for security and regulatory compliance," Analog Integrated Circuits and Signal Processing, vol. 69, no. 2-3, pp. 307–327, 2011.

[15] Measurement-Computing, "TracerDAQPro," Retrieved from: http://www.mccdaq.com/daq-software/tracerdaq-series.aspx, accessed: 2015-03-12.

[16] A. Mpitziopoulos, "PSU 101: A detailed look into power supplies," Retrieved from: http://www.tomshardware.com/reviews/power-supplies-101,4193.html, 2015.

[17] "Compact DC voltage and current sense PCB with analog output," Retrieved from: http://goo.gl/HPggRf, accessed: 2015-11-27.

[18] "Clonezilla," Retrieved from: http://clonezilla.org/, accessed: 2016-01-21.

[19] "Term0l5ter12.com," Retrieved from: http://www.urlvoid.com/scan/term0l5ter12.com/, accessed: 2015-11-19.

[20] "Malware Domains," Retrieved from: http://tamperdata.mozdev.org/source/browse/adblockplus/www/easylist/malwaredomains_full.tpl?rev=1.3.

[21] "How to get rid of TDL4 rootkit from Google chrome browser," Retrieved from: http://www.remove-browser-hijacker.com/en/How-to-remove-TDL4-rootkit-Redirect-Virus-from-Google-Chrome-20150526.html, accessed: 2015-11-29.

[22] E. Rodionov and A. Matrosov, "The evolution of TDL: Conquering x64," Retrieved from: http://go.eset.com/resources/white-papers/The_Evolution_of_TDL.pdf, accessed: 2015-11-29.

[23] Microsoft-Malware-Protection-Center, "Trojan:Win32/Alureon.FE," Retrieved from: https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?name=Trojan:Win32/Alureon.FE, accessed: 2015-11-09.

[24] "TDL4 Carrier to Glupteba," Retrieved from: https://goo.gl/NYnWGT, accessed: 2015-11-19.

[25] M. Rouse, "TDL-4 (TDSS or Alureon) definition," Retrieved from: http://searchsecurity.techtarget.com/definition/TDL-4-TDSS-or-Alureon, accessed: 2015-03-12.

[26] "Backdoor.Tidserv," Retrieved from: https://www.symantec.com/security_response/writeup.jsp?docid=2008-091809-0911-99, accessed: 2015-11-09.

[27] Remove-Malware-Tips, "Tips to get rid of rootkit.boot.pihar.b trojan," Retrieved from: https://www.removemalwaretip.com/windows-10/rootkit-boot-pihar-b-removal-technique-tips-to-get-rid-of-rootkit-boot-pihar-b-trojan.