Towards an Intrusion Detection System for Battery Exhaustion Attacks on Mobile Computing Devices

Daniel C. Nash, Thomas L. Martin, Dong S. Ha, and Michael S. Hsiao
Virginia Tech, Dept. of ECE
{dnash, tlmartin, ha, hsiao}@vt.edu

Abstract

Mobile computers are subject to a unique form of denial of service attack known as a battery exhaustion attack, in which an attacker attempts to rapidly drain the battery of the device. In this paper we present our first steps in the design of an intrusion detection system for these attacks, a system that takes into account the performance, energy, and memory constraints of mobile computing devices. This intrusion detection system uses several parameters, such as CPU load and disk accesses, to estimate the power consumption using a linear regression model, allowing us to find the energy used on a per process basis, and thus identifying processes that are potentially battery exhaustion attacks.

1. Introduction

A key element in a successful pervasive computing environment is a personal computing device that enables the user to have continuous access to information [7]. Users' reliance on these devices necessitates that they be secure. One security attack that is unique to battery powered devices is a denial of service attack aimed at draining the battery. These "sleep deprivation torture" or "battery exhaustion" attacks, as called by Stajano and Anderson, prevent devices from entering normal low power idle or sleep states [8]. Consequently, the expected battery life of the devices is greatly reduced and users fail to gain the full utility of the devices.

Battery exhaustion attacks are no longer theoretical. In our previous work, we have identified and implemented three different classes of these attacks [6]: (1) malignant attacks, in which a virus or Trojan horse is used to make the device consume significant power, (2) benign attacks, in which an unmodified program is given pathological data such that the program consumes excessive energy, and (3) service request attacks, a special form of the benign attack in which repeated requests are made to a network service provided by the device. The malignant attacks can be found using currently available virus scanning techniques, but the benign and service attacks cannot be detected with them because they work on unmodified code. In addition to our proof-of-concept implementations, there is already a virus "in the wild" that has the properties of a battery exhaustion attack, although its excessive power consumption appears to have been a side effect rather than the main intent. The Cabir virus was created to illustrate a vulnerability in mobile devices running Symbian OS Series 60 [9]. It transmits itself using the Bluetooth communication protocol between devices. While the goal of this virus writer does not appear to be the creation of a power attack, the operation of the virus causes one to occur. The virus causes the Bluetooth radio on the mobile device to broadcast at frequent intervals, seriously reducing the battery life of the device.

To combat these new attacks, a new line of defense must be developed and put into place. We propose the development of an intrusion detection system designed to detect this new form of attack, subject to the performance, memory, and energy limitations of pervasive computing devices. Section 2 of this paper discusses the design issues in creating such a system. Section 3 discusses the development of one system and its effectiveness while section 4 describes our conclusions and future work.

2. IDS Framework

The problem of intrusion detection (ID) has been studied for several years with early papers on the subject appearing in the later 1970s and early 1980s [3]. While the definition of an intrusion varies slightly from paper to paper, definitions such as the following are widely accepted: "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource" [5]. An intrusion detection system (IDS) then is a system which attempts to detect and in some cases react to intrusions, whether on one system, a group of systems, or a computer network.

The constraints placed upon the IDS of battery operated devices are much more severe than those placed on traditionally studied and commercial IDSs in existence today. Compared to a desktop or enterprise computer system, there are a large number of restrictions imposed on any IDS that can be deployed on most mobile devices. These restrictions include limited processor power, memory, and power usage. Whereas a typical desktop system today may have one to two gigabytes of memory, a typical iPAQ has 64 megabytes of main memory available. Likewise, while desktop systems have a large amount of processing power available with processors operating as high as the 3 GHz range, most mobile devices have processors operating at just a few hundred megahertz. A detection system implemented on these mobile devices must have a small footprint and limit the amount of power it consumes. Any solution to the problem of intrusion detection and specifically to the problem of detecting power attacks should not itself incur a large power and performance requirement.

Methods used in commercial IDSs may be difficult, if not impossible to implement. For instance, the extensive audit data, in the form of system logging, collected and analyzed by an IDS may simply not be present or may require too much time and energy to collect. Also, extensive analysis of this data may consume too much power or make a system too unresponsive for a user for it to be worthwhile. Most network IDSs also rely upon the cooperation of several detectors to gather enough information to cover the entire network. Most mobile devices operate independently.

The goal of an IDS that detects power attacks must be to identify attacks that cause the system to consume too much energy. Although it is impossible to prevent the attacks from using any energy, we believe the amount of energy consumed can be mitigated. Consequently, our goal is to try to guarantee a specific percentage of the overall battery life of the system. If a device could operate at idle power for 3 hours under normal usage, our goal might be to guarantee 2 hours of operational life in the face of repeated attacks. Given this goal, it is necessary to know when the system has high power consumption over a long period of time, such that the system is in danger of not meeting the guaranteed battery life. When the time threshold has been exceeded, we then identify which process or processes are responsible for using the most energy over that period of time. Such an IDS is unique in that it can still be successful even if it does not detect all attacks against the system. It allows attacks through that do not cause the system to exceed its energy consumption threshold. In that case, even though the attacks are successful, the goal of guaranteeing a specific battery life is still achieved.

2.1 IDS Parameters

The most straightforward way to detect a power attack would be to measure the power on a process-by-process basis, thus determining which processes were responsible for consuming large amounts of energy. Unfortunately, most, if not all, battery powered devices lack a self-contained, high fidelity power measurement system. If "smart battery" technology is used, rough measurements of power consumption and remaining battery capacity can be obtained, but these power measurements are too coarse to provide energy consumption on a process-by-process basis. Smart battery chips have low sampling rates, on the order of 1 Hz. To increase accuracy in estimating the remaining capacity, some of these chips also only report a value for the current power dissipation rate averaged over tens of seconds. These two properties of smart battery chips make it difficult to use them for accurate power measurements on a process-by-process basis.

Secondly, even if the battery operated devices of interest had high frequency power measurement systems available, use of such a system could leave a user susceptible to a key cracking technique known as power analysis [4]. For devices that perform encryption, the energy consumption can be correlated to bits of the encryption key, allowing an attacker to rapidly find large portions of an encryption key. If detecting battery exhaustion attacks requires a high resolution power measurement system, it could make this attack possible without requiring physical access to the device. Thus, we desire a method of estimating power on a process-by-process basis that does not have a high enough fidelity to make power analysis attacks feasible.

Barring direct power measurement, it is necessary to measure other indirect indicators of power usage on the system. Earlier work on developing different forms of power attacks have shown several components that can cause significant elevated power consumption on mobile systems [6]. Extended processor usage and repeated wireless transmission both caused extended elevated power levels in the devices tested. Repeated hard disk access or causing the hard disks to spin down and up repeatedly could also cause elevated power levels. As an example, Figure 1 illustrates the close correlation between processor usage and power consumption on an IBM Thinkpad T23 running Windows 2000 Professional.

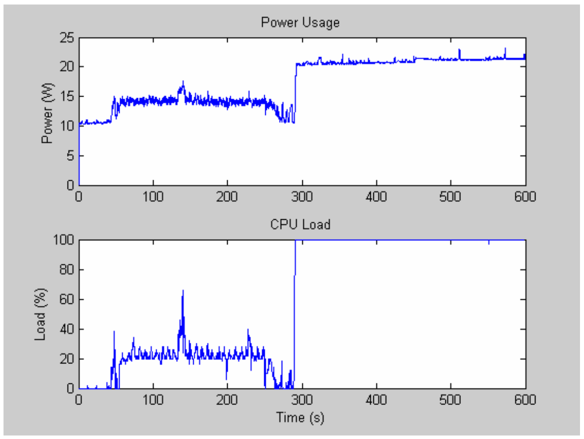


Figure 1 - CPU load (bottom) has a direct affect on system power consumption (top).

3. A Power Attack IDS

3.1 Power Estimation

The correlation between CPU load and power consumption gave rise to the idea of predicting the power consumption of the overall system based on various system metrics, including CPU load, disk read and write accesses, and network transmits and receives, using a linear regression model. Using the Microsoft Performance Data Counters available in Microsoft Windows NT 4.0 and later operating systems, many of these metrics were measured on the above mentioned system while at the same time measuring the power consumption of the system externally. The power measurement setup used was the same as that used in [6], using a high-end digital multimeter capable of sampling system current at 10,000 samples per second. The laptop was chosen for this paper because it provided greater flexibility in testing our fundamental ideas than would a more limited platform such as a cell phone, but we believe the method we describe here can be generalized to work for any battery-powered computing system with the appropriate choice of variables for the regression model. The system metrics for the laptop that were chosen to be monitored included the following: percentage of time the processor was busy with non-idle threads; percentage of time spent doing physical disk reads; percentage of time spent doing physical disk writes; the number of bytes per second doing network receives; the number of bytes per second doing network writes; and the number of memory-page faults per second. Many of the other possible metrics the Performance Data Counters monitored were tied to the operating system itself such as the file system cache and print queue. The selected parameters gave information on physical devices in the system that were suspected of influencing power consumption.

Multiple linear regression was used to find the correlation coefficients for each of the measured metrics. The β_0 parameter encompasses the power consumption of devices on the system such as the display, CPU fan, and other device metrics not specifically monitored. Using these coefficients, shown in Table 1, the following equation was used to calculate the estimated power usage of the system:

$$\begin{aligned}\text{Estimated Power} = {} & \beta_0 + \beta_1 * (\%\text{CPU Load}) + \beta_2\,(\%\text{Disk Reads}) \\ & + \beta_3 * (\%\text{Disk Writes}) + \beta_4 * (\text{Network Bytes Read/Sec}) \\ & + \beta_5 * (\text{Network Bytes Written/Sec}) + \beta_6 * (\text{Page Faults/Sec})\end{aligned}$$

Equation 1 - Power estimation from system metrics.

Figure 2 compares the resulting power estimation with the actual power measurements while Figure 3 shows the error of the power estimation. The mean error in the estimation is 5.67 %. Most of the large deviations occur at the places where the power transitions are very large, which is most likely due to difficulty in measuring the system metrics and power usage at exactly the same time

Coefficient	Value
β_0	11.076 W
β_1	0.0897 W/(% CPU load)
β_2	0.0207 W/(% time servicing disk read requests)
β_3	0.0126 W/(% time for disk write requests)
β_4	4.478×10^{-6} W/(# of network bytes received)
β_5	-6.382×10^{-5} W/(# of network bytes sent)
β_6	-4.123×10^{-5} W/(# of memory page faults)

Table 1 - Multiple linear regression was used to derive these coefficients.

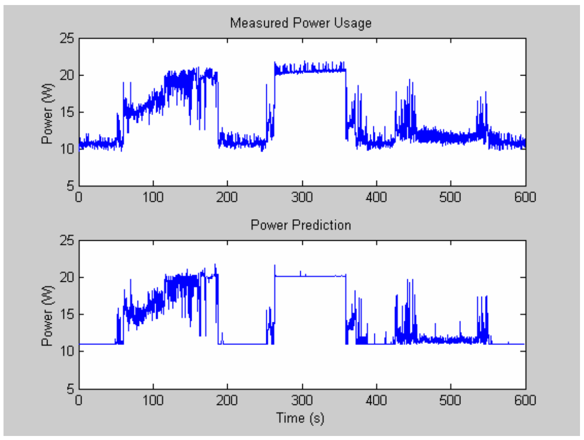


Figure 2 - Comparison of power estimation (bottom) with actual power usage (top).

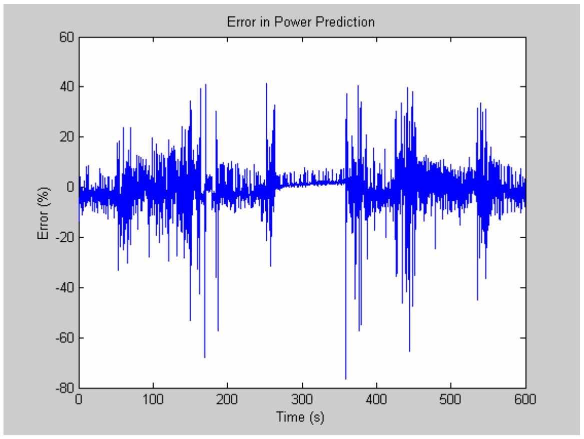


Figure 3 - Error in power estimation.

instants. Averaging the sample would likely reduce this error, but also hinder the ability to estimate large changes in power usage more quickly.

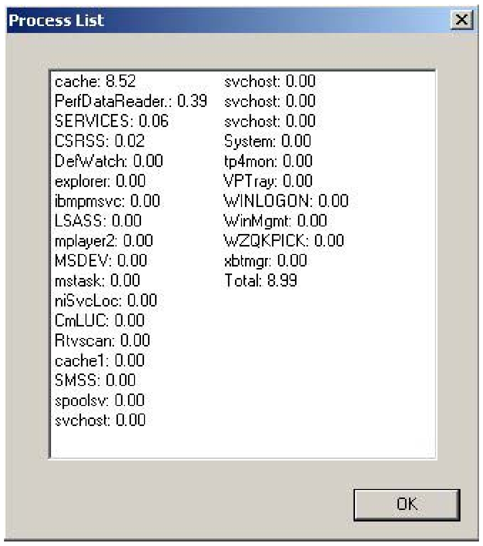
Those errors, while undesirable, do not prevent the IDS from using the power estimation. Determining the instantaneous power of the system is not necessary. The IDS must detect and possibly react to elevated power consumption over the threshold discussed above as that level of power will prevent the system from meeting its goal. Over the time period that the IDS monitors, the average error in the estimation is much lower, permitting an accurate estimate of the system's power usage.

3.2 Process Identification

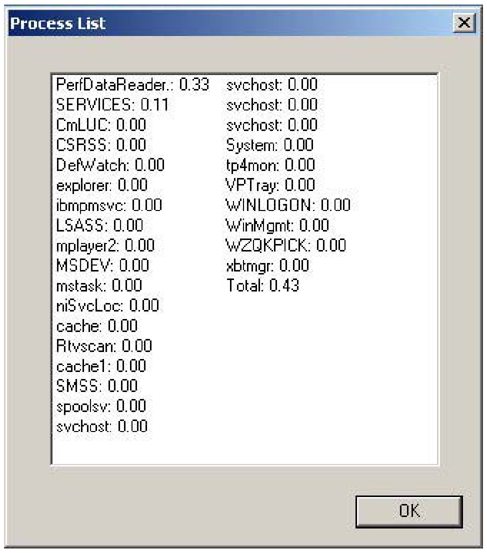
With the power estimation, it can be determined when the system has exceeded a given power threshold for an extended period of time and the goal of guaranteeing a certain battery life cannot be achieved. The task is then to determine what processes are causing the increased power consumption. While all the metrics used in estimating the power cannot be easily determined on a per process basis, the amount of processor usage can be. From the linear regression, processor usage proved to be the largest factor in power consumption. So, using the processor usage of each process as a means of determining its affect on overall system power usage is a good starting point. The ratio of a process' processor usage to the overall system usage provides the measure for what that process contributes to the overall system power consumption. Such a method was used to generate the list shown in Figure 4, which shows a power-ranked list of all the processes running on the system when it is under attack and when it is not. In the top figure, a malignant power attack named "cache" is running on the system, while in the lower figure, the system is operating normally. The attack is clearly distinguishable.

The power estimates for each process are an average of the power each process used over a 5 second window. This window was chosen as a starting point to observe long term process behavior. The size of the time window used could be shortened to make the IDS more sensitive to sudden increased power usage on the part of a process or lengthened to capture longer term power increases. The best choice for the window of time may be dependent on what forms of attack a device is most sensitive. Care must also be taken in selecting a window size that will not trigger the IDS for short term power increases caused by legitimate non-attacking programs, called a false positive. Reducing the false positive rate is one of the top priorities of IDS designers as a high false positive rate hinders the ability to capture actual attacks and may cause users to turn off the IDS system.

Estimating power consumption system wide and on a per process basis as a detection method is well suited for the constrained environment of mobile devices. The computational effort is relatively low, requiring only a few floating point operations on the collected data to make an estimate. This is especially true when compared to other detection methods such as signature detection, which can require a large database of signatures and a large number of comparisons to make a decision. The memory footprint of the IDS, as it is currently implemented, is also very small since it eliminates the need to save large quantities of data.



```
Process List
cache: 8.52                 svchost: 0.00
PerfDataReader.: 0.39       svchost: 0.00
SERVICES: 0.06              svchost: 0.00
CSRSS: 0.02                 System: 0.00
DefWatch: 0.00              tp4mon: 0.00
explorer: 0.00              VPTray: 0.00
ibmpmsvc: 0.00              WINLOGON: 0.00
LSASS: 0.00                 WinMgmt: 0.00
mplayer2: 0.00              WZQKPICK: 0.00
MSDEV: 0.00                 xbtmgr: 0.00
mstask: 0.00                Total: 8.99
niSvcLoc: 0.00
CmLUC: 0.00
Rtvscan: 0.00
cache1: 0.00
SMSS: 0.00
spoolsv: 0.00
svchost: 0.00
```



```
Process List
PerfDataReader.: 0.33       svchost: 0.00
SERVICES: 0.11              svchost: 0.00
CmLUC: 0.00                 svchost: 0.00
CSRSS: 0.00                 System: 0.00
DefWatch: 0.00              tp4mon: 0.00
explorer: 0.00              VPTray: 0.00
ibmpmsvc: 0.00              WINLOGON: 0.00
LSASS: 0.00                 WinMgmt: 0.00
mplayer2: 0.00              WZQKPICK: 0.00
MSDEV: 0.00                 xbtmgr: 0.00
mstask: 0.00                Total: 0.43
niSvcLoc: 0.00
cache: 0.00
Rtvscan: 0.00
cache1: 0.00
SMSS: 0.00
spoolsv: 0.00
svchost: 0.00
```

Figure 4 – Process list for system under attack by a virus called *cache* (top) and with only IDS active (bottom).

4. Conclusions and Future Work

The proposed IDS proved to be effective in identifying when the system has exceeded the power consumption that would allow it to achieve a guaranteed battery life. Our IDS can also identify those processes that caused the increased load on the system. This allows a user to take a necessary action to stop such processes from continuing to operate. They may also allow the action to continue if it is a process they want to continue to run, such as a virus scanner.

Increasing the accuracy of the power estimation may still be accomplished through measurements of other system metrics. The cache attack program used in [6] showed increased power consumption for cache misses on some platforms while other platforms showed increased power consumption for cache hits. Through mechanisms such as Intel's Pentium performance counters, cache behavior could be recorded and integrated into the power estimation.

As it stands now, the developed IDS is still vulnerable to power attacks that distribute the work of the attack across multiple processes. An attack that spawns multiple processes where each process contributes to a small portion of an overall attack before ending, would make it difficult to isolate any process as an attack. However, methods of determining what processes spawned other processes are available and would make it possible to classify the behavior of multiple processes acting in concert. Ideally, we would like to make the IDS resistant to this type of attack.

Making the IDS reactive to attacks instead of just identifying them would also improve the effectiveness of the system. In work presented in [1], system-call delays were used to mitigate the effect of programs that were suspected as anomalous by reducing the rate at which they operated. A similar technique, perhaps in the way the suspected program is scheduled by the operating system, could be used to reduce the power consumption of the system back to a level that would achieve the guaranteed battery life of the system. This method could even be adaptive to allow for more power consumption by the system as the guaranteed battery life is neared.

Furthermore, determining the energy consumed by each process could be used to trigger a more complex IDS. Rather than having a more complex IDS running all the time, and thus consuming precious energy, the process energy estimation could serve as a first line of defense. When the system energy consumption becomes high enough, then the more complex IDS could be used to analyze the system state to determine if the behavior truly indicates an attack or is normal. An open question is whether an IDS based on self-contained power measurement such as we have described here can be used to detect non-battery-related intrusions. Given the power consuming side effects of the Cabir virus [9] and our own experience with viruses on laptop computers, there is good reason to believe that power consumption information can augment existing techniques for intrusion detection, e.g. [2].

Finally, it is necessary to develop a methodology for determining the parameters used for the linear regression such that power can be estimated adequately for the wide variety of battery-powered systems that we expect will be used in a pervasive computing environment. Our current regression model uses disk accesses, for example, which is not a factor for most PDAs and cell phones. This could become part of the design process of devices, although some allowances should be made for configurations of individual devices (e.g., amount of memory and PCMCIA cards).

Acknowledgements: This material is based upon work supported by the National Science Foundation under grant no. ANI-0219801.

References

[1] S. Forrest and A. Somayaji, "Automated Response Using System-Call Delays," *Proceedings of the 9th USENIX Security Symposium*, pp. 185-197, August 2000.

[2] S. Forrest, S. Hofmeyr, and A. Somayaji, A. "Computer immunology," Communications of the ACM, , October 1997, vol. 40, no. 10, pp. 88-96.

[3] A. Jones and R. Sielken, "Computer Intrusion Detection: A Survey", University of Virginia, Computer Science Technical Report, 2000.

[4] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," Advances in Cryptology, Crytpo '99, Springer LNCS 1666, pp. 388-397, 1999.

[5] W. Lee and S. Stolfo, "Data Mining Approaches to Intrusion Detection," *Proceedings of the 7th USENIX Security Symposium*, pp. 79-94, January 1998.

[6] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-Service Attacks on Battery-powered Mobile Computers," *Second IEEE International Conference on Pervasive Computing and Communications*, pp. 309-318, March 2004.

[7] M. Satyanarayanan, "Pervasive computing: vision and challenges," IEEE Personal Communications, Volume: 8 Issue: 4, pp. 10 -17, Aug. 2001.

[8] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for adhoc wireless networks," in *Proceedings of the 7th International Workshop on Security Protocols, Lecture Notes in Computer Science volume 1796*, pp. 172-194, April 1999.

[9] Symantec Corporation, "SymbOS.Cabir," http://securityresponse.symantec.com/avcenter/venc/data/epoc.cabir.html.