Denial-of-Service Attacks on Battery-Powered Mobile Computers

Jayan Krishnaswami

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. Thomas L. Martin, Chair
Dr. Michael S. Hsiao
Dr. Dong S. Ha

February 13, 2004
Bradley Department of Electrical and Computer Engineering
Blacksburg, Virginia

Keywords: denial-of-service, battery, mobile devices

Denial-of-Service Attacks on Battery-Powered Mobile Computers

Jayan Krishnaswami

ABSTRACT

A Denial of Service (DoS) attack is an incident in which the user is deprived of the services of a resource he is expected to have. With the increasing reliance on mobile devices like laptops and palmtops, a new type of DoS attack is possible that attacks the batteries of these devices, called "sleep deprivation attacks". The goal of sleep deprivation attacks is to rapidly drain the battery of the mobile devices, rendering the device inoperable long before the expected battery lifetime, thus denying the service the user expects from the mobile device. The purpose of this research is to investigate these types of attacks so that proper defense mechanisms can be put in place before the attacks become a more sophisticated and potent force. This research presents three different possible methods that can be adopted by an attacker to drain the battery of a device i.e. malignant attacks, benign attacks and network service request attacks. These attacks are implemented on a variety of mobile computing platforms like palmtops and a laptop and the corresponding results are presented. Finally, a mathematical model is presented that estimates the battery life of a device based on its power consumption in various power management states and expected usage. This model can also be used to predict the impact of a DoS attack on the battery life of the device under attack.

Dedicated to my parents and my sister.
I would not be who I am
without their support and blessings.

Acknowledgements

I am extremely grateful to Dr. Thomas L. Martin for the immense support and encouragement he gave me during my term as a graduate research assistant. His insightful advice and guidance cleared a lot of hurdles I came across during my research.

I am extremely thankful to Dr. Michael S. Hsiao and Dr. Dong S. Ha for their help and guidance, and their willingness to serve on my committee. Their suggestions during the weekly meetings helped a great deal in making my work a success.

Special thanks to all my friends in Blacksburg who have made my stay here a very fruitful and enjoyable one. I will always cherish the time we spent together.

Finally, I would like to thank my family for their unfailing love and support throughout my career. Without their continued backing and blessings, this effort would not have been possible.

Jayan Krishnaswami

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

Battery-powered mobile devices like laptops and palmtops have been gaining in popularity over the past few years. Laptop and palmtop sales have been increasing steadily, reaching record numbers during the last year [22, 23]. There is an increasing reliance on these devices in education [25], healthcare [24] and defense [26]. The increasing dependence of users on these mobile devices requires that these devices be made more robust and resilient against different types of security threats that evolve over time and target subsystems not targeted before. In this regard, a new type of "Denial of Service" attack has been created. These Denial of Service (DoS) attacks target the battery of the mobile devices, rendering the device inoperable long before the expected battery lifetime, thus denying the user the service it expects from the mobile device.

1.1 Motivation

A typical mobile computer is expected to give a certain battery life time assuming that the user is actively using the device for a fixed amount of time and the device is in idle mode for the remaining time. Power management schemes tend to put the device in low power sleep or standby modes if the device is idle for some amount of time. In these

conditions if an attacker can cause the device to execute power hungry applications so that the device is not able to enter into a low power sleep mode, the battery life of the device would be cut down drastically. Stajano and Anderson have called these attacks as "*sleep deprivation torture*" attacks [7].

Denial of service attacks that clog network traffic or use up resources on web and email servers require that the attacker maintain the attack as long as he wants other users to be denied the service of the servers. However DoS attacks on batteries do not require the attacker to keep up the attack because, once the battery has been drained, the attacker can move on to attack other systems.

With the development and advancement of computer networking, viruses, Trojan horses and other forms of security attacks became more prevalent and sophisticated. In the same way, the increasing use of battery-powered mobile devices in day-to-day activity could potentially lead to increasingly enhanced and competent DoS attacks on batteries. A lot is known about attacks on vulnerabilities of mobile devices that they share with other types of computers, and ways of preventing them. However not much is known about sleep deprivation attacks. This makes the mobile devices particularly vulnerable to a DoS attack on their batteries.

As described in Chapter 2, three methods of DoS attacks are envisioned to drain the battery:

- Malignant attack: application code or kernel binary is modified to increase power consumption
- Benign attack: the device is made to execute a valid but energy-hungry application without modifying the application
- Service Request attack: repeated requests are made to the device over the network

Based upon an analysis presented in [12], an attacker could potentially cut down the battery life of the device under attack by a factor of 30 to 280. Considering the enormity of the impact of these attacks, it is imperative to consider the battery of a mobile device as a vulnerability and hence, methodologies should be devised to protect it.

1.2 Contributions of this thesis

This thesis explores the effect of denial of service attacks on batteries of mobile devices. With time it is expected that these attacks would become more sophisticated, potent and increase their impact on reducing the battery life. Hence, it is all the more necessary for developers to incorporate methods of preventing these attacks during design time rather than patching up the system once the attack has been made. The research contributes the following:

- One of the goals of this thesis is to raise awareness of the impact of denial of service attacks on the batteries of mobile devices so that a defense mechanism can be put in place much before such attacks become widespread.
- Different forms of these attacks were implemented on a variety of platforms like palmtops and a laptop to study the impact on increase in power consumption caused by the attacks. These are the first real examples of attacks on batteries of mobile computing devices.
- In order to help developers estimate the impact of sleep deprivation attacks on the battery lifetime a battery usage model is proposed. This model takes into account the power consumption values of a device and based on typical usage characteristics, gives an estimate of the battery life of the device.
- The model also provides an estimate of the effect a denial of service attack can have on a device's battery life. As will be shown in Chapter 4, a laptop under a malignant attack could have its battery life reduced by a factor of two under normal usage while an iPAQ could have its battery life reduced by a factor of five.
- This model can also be applied to classes of systems with similar vulnerabilities to gauge the effect of a denial of service attack on the systems' battery life. Such a model would hence help developers in designing systems with an eye towards possible sleep deprivation attacks and the impact of these attacks on the battery life.

1.3 Thesis Organization

This thesis is organized in the following manner. Chapter 2 gives a brief introduction to existing Denial of Service attacks and their different forms. Denial of service attacks on batteries is a known issue for sensor networks. The chapter also explains the security concerns of sensor network developers with respect to DoS attacks on batteries. Finally, a classification of the possible methodologies adopted by an attacker to implement a DoS attack is also provided in this chapter. Chapter 3 shows some results of example DoS attacks implemented on a laptop and on PDAs. The experimental setup used and the procedure for implementing the attacks is also explained. Chapter 4 introduces a battery usage model for estimating the battery life of different devices based on the certain parameters. The effect of DoS attacks on the battery life of devices is then estimated using this model. Chapter 5 summarizes this thesis and also discusses avenues for future research.

Chapter 2

2 Background

This chapter provides the background information necessary for understanding Denial of Service (DoS) attacks. It begins with a brief description of DoS attacks and their different forms. This chapter also gives some information on the security concerns of sensor networks and other wireless adhoc networks with respect to DoS attacks. This chapter also describes a new type of DoS attack that takes advantage of loopholes in algorithms of certain applications on mobile devices. Finally, an introduction is made to the classification of the potential forms of DoS attacks on mobile devices.

2.1 Denial of Service Attacks

A Denial of Service (DoS) attack is an incident in which a user is deprived of the services of a resource he would normally expect to have [31]. The most famous form of DoS attacks is the one made on different web servers connected to the Internet. For example, the DoS attacks made on popular websites like Yahoo, CNN, eBay and Buy.com in February 2000 caused considerable damage in terms of revenue [13] because of loss of earnings when the web servers were knocked offline for about three hours.

A very common form of DoS attacks involves sending a large number of common packets aimed at a single destination. The most common packets used are:

- TCP
- ICMP (echo request/reply)
- UDP

The huge traffic deluge caused by these packets leads the network to no longer be able to distinguish between legitimate and malicious traffic. Basically all available bandwidth is used up and nothing is left for legitimate use causing the users to be denied the service of the network. Web servers are often targeted for DoS attacks by making a large number of HTTP requests to the servers. In most cases these attacks are made more effective by changing a few attributes on the packets. For example, false source IP addresses could be embedded into the packets (IP Spoofing) so that the server can't find the sender when it is trying to communicate.

Another variant of DoS attacks is the Distributed Denial of Service (DDoS) attack. These attacks break into other computers and use all the compromised computers to collectively launch a DoS attack on a system or server. Such attacks greatly multiply the effect of a normal DoS attack. This mode of attack was used in the attacks on the servers of popular websites mentioned earlier.

Defending against DoS attacks has a more reactive approach rather than a proactive one [28]. Because of the advent of IP Spoofing distributed attack methods, it has become increasingly difficult to prevent these attacks from occurring. In most cases, once the attack has occurred, the source (or sources) of attacks are identified and shut down. A firewall with an intrusion detection system that looks for specific traffic patterns can be a first step to lower the risk of DoS attacks. Rate limiting and packet filtering are other options that can help reduce the impact of DoS attacks.

The DoS attacks explained so far are related to servers providing a wide variety of services to a large number of users. Moving away from servers and the internet and coming towards day-today battery powered mobile computing devices like laptops, palmtops and cell-phones, a new type of Denial of Service attack can be envisioned. If these devices are made to execute an invalid energy hungry program repeatedly, it will cause their battery to drain out much before their expected life time. This would cause

the user to be denied the service of his battery powered mobile device. Also, unlike the DoS attacks mentioned earlier, the attacks on the batteries of mobile devices do not require the attacker to keep up the attack to deny service. Once the battery has been drained, the attacker can move on to attack other devices. The next section looks into some of the DoS attacks on batteries for sensor networks.

2.2 Wireless Sensor Networks

Wireless sensor networks comprise of a large number of application specific sensor nodes deployed over an area of interest. The sensor nodes communicate with each other wirelessly and work together towards a common goal of sensing, collecting and processing data. In this way, sensor networks provide a low cost option for applications like military surveillance [14], healthcare [15, 16] and environmental studies [17] by forming ad hoc relationships with little or no existing infrastructure. At such vital and security-sensitive deployments, it is essential that the sensor networks are always up and running and performing the tasks meant for them. If these networks are rendered inoperable because of Denial of Service attacks, it could risk damage to the health and safety of people [6].

As explained earlier, sensor networks are meant to function in harsh environments without human supervision. Hence, battery resources are at a premium. Practically it may not always be possible to replace the battery of a sensor node. It is in these conditions that a DoS attack on the battery can prove to be detrimental to the functionality of sensor networks.

The remainder of this section looks into the issue of DoS attacks for sensor networks.

2.2.1 Denial of Service attacks on Wireless Sensor Networks

One of the earliest mentions in the literature regarding rendering a mobile device inoperable by draining its battery is made by Stajano and Anderson in [7]. Nodes often go to sleep during periods of inactivity only to wake up once in a while to listen to

incoming radio signals. This helps a great deal in reducing the power consumption of the nodes. It is in these scenarios that an attacker can prevent the node from going to sleep by constantly engaging it in traffic, and thus causing the battery resources to die out. These attacks have been termed "sleep deprivation torture" attacks [7].

In order to prevent unauthorized access to the network, authentication can be provided using cryptographic algorithms. But the protocols incorporating these algorithms must be efficient enough to not become the target of battery exhaustion attacks themselves. Aura et. al. provide a method to use "client puzzles" as a first step to authenticating a client before using up resources in a cryptographic algorithm [8]. The rule of thumb of this principle is that before entering into any sort of reliable communication, the expenses incurred by the client (node initiating communication) should be greater than that of the server. The expenses, in terms of resources used, will be increased on the client side when it is asked to solve a puzzle put forth to it by the server. Once the server verifies the client's solution to the puzzle, it can perform cryptographic operations to authenticate the client.

Although there are many factors (software and hardware bugs, environmental conditions) that could diminish the capacity of the network to provide the requisite service, Stankovic and Wood particularly look into protocol level vulnerabilities [6]. It enlists the layered network architecture of the sensor networks, the denial of service vulnerabilities of each layer and the defenses possible against these attacks. Of particular interest in these attacks is the one mentioned for the Medium Access Control (MAC) layer. The main function of the MAC layer is to allocate channel resources fairly among nodes for communication. This prevents two or more nodes from using the channel concurrently and causing collisions. Collisions are known to be a major cause of energy wastage because the corrupted packets have to be discarded and the ensuing retransmissions increase energy consumption. If the MAC layer is not optimally designed then it could cause the transmitting node to continuously retransmit a frame, causing the battery resources of the nearby nodes to get depleted.

Another variant of this attack is called the "interrogation attack" [6], which makes use of the interaction that takes place between two nodes prior to data transmission. For example, wireless LANs (IEEE 802.11) use Request To Send (RTS) and Clear To Send

(CTS) messages to request channel access. A node under attack could constantly send RTS evoking a CTS response from the receiver and thus using up battery resources at both ends. A possible solution to these battery exhaustion attacks would be to make the MAC layer "rate limiting" [6], which would cause the network to ignore requests beyond a particular threshold value.

One thing to note about the denial of service attacks mentioned for sensor networks is that the solutions often tend towards trying to reduce the traffic. This is because the major source of power consumption in sensor networks is the RF subsystem. Hence if the traffic is reduced, the power consumption is bound to go down and thus the service of the sensor networks would not be denied before its normal life cycle. However when considering general purpose mobile computing devices, reducing the traffic is not a viable option because they provide the user with many more avenues of usage compared to sensor networks. Also, unlike sensor networks, communication does not always dominate the power consumption, as will be shown in Section 3.3.3. In addition to communication, CPU utilization and screen updates are major sources of battery loads. Hence, an attacker can use any of these subsystems to cause a denial of service power attack on the mobile device. Thus, it becomes necessary to study the types of attacks possible on general purpose mobile computing devices as a first step to try to guard against them.

The forms the attacks can take are Malignant Attacks, Benign Attacks and Service-Request Attacks, which will be described in detail in Section 2.4. Before discussing these forms of attacks, however, the next section describes a DoS attack that could be used to implement either Benign or Malignant power attacks, the algorithmic complexity attacks.

2.3 Denial of Service via Algorithmic Complexity Attacks

Crosby and Wallach demonstrate in [1] that the running time of certain applications can be brought up to their worst case by providing them with malicious inputs. They make use of the fact that common applications use data-structures and algorithms that

have an average running time much less than the worst case, expecting the worst case behavior never to occur. Results shown in [1] prove that the malicious data generated can cause a dedicated Bro server (a general purpose network intrusion detection system) to drop 71% of its traffic because its CPU is being consumed in dealing with the malicious data.

Hash table vulnerabilities are particularly targeted in these types of attacks. In order to understand the hash attack, a brief explanation of how objects are inserted into hash tables is provided. An object to be inserted into a hash table is first associated with a *hash value*. The size of the array into which these objects are to be inserted is called the *bucket count*. The hash table uses a *hash function* to insert the object at a position in the array equal to the hash value modulo bucket count. If two objects map to the same position in the array, it is called a *collision*. One possible solution to resolving collisions is to let the bucket form a linked list of all objects mapped to that position in the array. These linked lists are called *hash chains*. Knowledge of how a particular program inserts data into the hash table can lead an attacker to maliciously modify the input data so that every object maps to the same position in the array causing the hash chain to become very long. Each time a new object is to be inserted into the hash table, the hash function will scan every entry in the bucket to check for duplicates; thereby executing the worst case performance.

To illustrate the enormity of this type of attack, Crosby and Wallach created malicious input data for two versions of Perl (5.6.1 and 5.8.0) having different hash functions. When an interpreter was given input meant to cause collisions, the time taken to load the input was around three orders of magnitude worse than the time taken to load normal input (~2 seconds compared to almost 2 hours) [1].

2.4 Potential Forms of Attacks

An attacker would ideally like to do negate the effect of the power management features on the device he is trying to attack. The attacker would target the subsystems that are major sources of power consumption because of the large difference in idle state

power consumption and active state power consumption. When considering a general purpose mobile computing device like a laptop or a palmtop, the major sources of power drain are the processor, the screen and the wireless LAN (WLAN) cards [18].

As regards the CPU power consumption, the power can be drastically reduced by scaling down the operating frequency and voltage on the fly [19]. For example, Intel Speed Step technology [9] uses frequency scaling in order to conserve power while giving optimum performance when laptops are running on batteries. This causes the power consumption to reduce by a factor of two in most cases as shown in [9].

Screen brightness has a direct relation to the power consumed from the battery. When running on batteries, power management dims the screen brightness in order to conserve energy [18].

Wireless LAN cards cause an additional load to be incurred by the battery. When a WLAN card is inserted into a laptop or a palmtop, the power consumption of the device increases not only because of the WLAN card itself, but also because of the power consumed by the rest of the platform in supporting that WLAN card [10]. Also, in the case of WLAN cards, the bulk of their power consumption is due to communication i.e. data transfer [20]. Approximate power consumption numbers for an Orinoco Silver WLAN card show that it consumes around 1.4 W in transmit mode, 0.9 W in receive mode and only 0.05 W in sleep mode [20]. Hence the power management features of WLAN cards put the card into sleep mode when there is no data to be transferred.

When trying to implement sleep deprivation attacks, the attacker will have to take into account the perceptive difference caused by the attacks on the functionality of the system as a whole. The attacker should be able to both enter the system unnoticed and execute the attack. Consequently, the attacker should take care to reduce the side effects that can be caused by these attacks. Some of the side effects that the attacker should take note of are:

- The CPU fan turns on although the user is not doing any CPU intensive activity.
- The system becomes less interactive than normal.
- The hard drive repeatedly spins up immediately after a spin down.

In spite of taking these measures, it may be possible for existing security techniques and antivirus software to detect these attacks. But there is also a possibility of these techniques falling prey to the sleep deprivation attacks. For example, an attacker could devise a virus that he knows would be caught by the anti-virus software but the energy spent in running a scan to detect it would cause a sleep deprivation attack.

The remainder of this section explains the classification of possible sleep deprivation attacks. The attacks are classified into malignant attacks, benign power attacks and service request attacks. The mechanism used by each of the attacks to drain the battery of the device under attack is different, but the goal is to keep the device under attack busy and prevent it from going into low power sleep modes.

2.4.1 Malignant Attack

A malignant attack is one in which an executable file is created or existing application code or kernel binary is modified to increase power consumption. This attack could also be implemented as a virus or a Trojan horse, a program which has malicious code hidden inside a seemingly harmless program that can do its chosen form of damage to the system.

The simplest malignant attack would be an infinite loop. But this would be very easily detected by the user because it would cause the system to become less responsive and perhaps cause the CPU fan to turn on. In order to escape notice of the user, the attack could be run at the lowest possible priority higher than the idle task. Then it would run only when the scheduler has no other task to run, thus preventing the device from becoming less responsive, but not allowing it to drop into low power idle state. Also, in order to prevent the CPU fan from turning on, the attack could also keep track of the CPU temperature and the algorithm used to keep the temperature in check. The attack may temporarily suspend its action when the CPU fan is about to turn on, remain idle to allow the CPU to cool down, and then resume its action. A drawback of this implementation is that it would drastically reduce the available idle time on the CPU and could hence be easily detected by a daemon that checks on idle time of the system.

A more sophisticated malignant attack would be able to patch the binary code of the operating system to modify its idle loop. Operating systems employ the use of an idle loop whenever there is nothing for the CPU to do. In this loop the CPU goes into a low power state by stopping the clock to the CPU, but the clock to the peripheral devices remains active. In most systems executing the idle loop requires only a few instructions. For the x86 architecture a single instruction ("HLT") needs to be executed while the StrongARM SA-1100 enters the idle mode if a sequence of three instructions are executed [11]. The attack could either prevent these instructions from being executed, or could patch the binary of the operating system in such a way that more power-hungry instructions are executed. This particular attack would be very hard to detect because the system response would not deteriorate and also there would be sufficient idle time left on the CPU making it difficult for the daemon to detect it. Of course an attacker who has gained sufficient privileges to modify the kernel file would likely deny service by more conventional means such as erasing files on the system.

2.4.2 Benign Power Attack

The main concept behind benign power attacks is that the attacker could cause the device to execute a valid but energy hungry task. Such attacks would be very hard to detect, but easy for an attacker to implement because the attacker would not have to make any changes to the software or hardware of the device under attack. All he needs to do is to provide data to the existing software on the device such that it would consume much more power than it would do under normal circumstances.

An example of such an attack is incorporating an animated GIF in a webpage that the user perceives to be unanimated. Another mode could be to execute a Java applet that is invisible to the user but drains the energy source.

The attacker could also provide the applications on the device under attack with inputs that cause it to consume much higher power than anticipated. As explained in Section 2.3, a possible avenue for this type of attack would be the exploitation of the hash algorithm in some applications.

2.4.3 Service Request Attack

Service request attacks include those attacks that could cause the power consumption of the device under attack to increase by engaging it in servicing invalid network requests. These attacks could be implemented by repeatedly making network requests like telnet, ssh or web requests to the device under attack, thereby draining its battery capacity.

As explained earlier, power consumption of WLAN cards is the maximum during data transfer. By constantly engaging it in traffic, the attacker would be preventing the WLAN card from going to sleep or idle mode and thus negating the effect of power management.

Another aspect of this attack would be the authorization. When the attacker would engage the device in requests, he could do so even with an invalid username/password combination. The processing used by the device to check the password would use up the battery energy. As an example, consider an attacker trying a service request attack on a battery-powered UNIX system with telnet requests and an invalid username/password combination. The telnet protocol is based on option negotiation. Each time a telnet request is made, options feasible to both the client and server are negotiated before a connection is made. Once the options have been decided, the client and server (device under attack) authenticate each other. If the authentication fails because of a wrong username or password, the client and server would have to renegotiate the options to setup a new connection. If the attacker could continuously repeat these steps then it would consume significant battery capacity on the server end.

Chapter 3

3 Power Attack Examples

As described in the previous chapter, there are three types of sleep deprivation attacks: Malignant power attacks, Benign power attacks and Service request attacks. This chapter gives examples of each of the three types of power attacks and includes experimental results of each attack on PDAs and a notebook computer. These examples are not meant to be exhaustive, but instead are intended to demonstrate that the attacks exist and that they have the potential to drastically decrease the battery life. Before describing the attacks and the experimental results in detail, a brief description of the experimental setup is given.

3.1 Experimental Setup

The setup for taking power measurements is shown in Figure 3.1. Data for the power consumption of the mobile devices was collected using an Agilent 3458A Digital Multimeter with a sampling rate of 10,000 samples / second. The multimeter was controlled by a computer over a GPIB cable. This computer was also used for storing the readings taken in an arrangement similar to that in [4]. In order to be able to take the

readings over multiple runs without having synchronization problems, the multimeter was made to start taking readings only after it received an external trigger from the device under test. A program was written for the laptop that 'sleeps' for a fixed amount of time before giving out a pulse on its parallel port that, in turn, was connected to the multimeter's external trigger input. This helped to begin the applications on the laptop within a couple of milliseconds of each other over many runs.

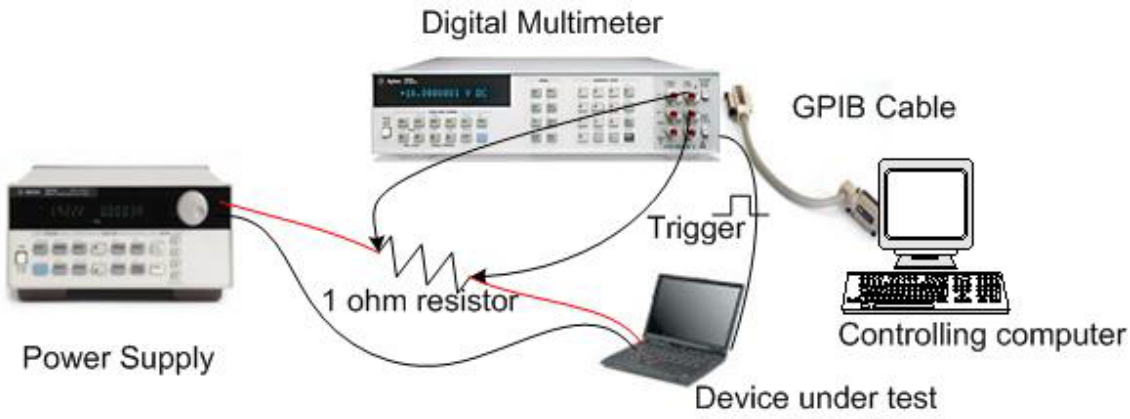

Figure 3.1: Setup for taking power measurements

For the measurements, a notebook computer and two different PDAs were used as the devices under attack. Results were measured on an IBM Thinkpad T23 notebook, a Compaq iPAQ model 3760 PDA and a Compaq Itsy, a research prototype PDA [5]. The IBM Thinkpad had an 866MHz Pentium III CPU, 128 MB of memory and dual booted the Windows 2000 and Linux operating systems. The iPAQ was a 3675 model, with 64 MB of memory, running Windows CE version 3.0. Another iPAQ was used to implement Service request attacks described in Section 2.4.3. This iPAQ was a 3630 model, with 32 MB of memory and running Familiar Project Linux v0.5.3. The Itsy was a version 1.5, with a 206 MHz StrongARM processor and 64 MB of memory, running Linux. Both the IBM Thinkpad and the iPAQ had PCMCIA slots, and an Orinoco Silver 802.11b wireless network card was used to provide a network connection to them for the service request attack results.

For the Thinkpad and the iPAQ, screen brightness was a large factor in power consumption. To provide consistent behavior over multiple trials, the screen brightness

was set to its minimum value during all measurements. The power management settings for the Thinkpad and the iPAQ were the same for each trial.

The plot of power consumption of the Thinkpad when under a malignant attack (the attack implementation will be explained in the next section) is shown in Figure 3.2. The X-axis denotes time in seconds while the Y-axis denotes power in watts. As shown in this figure, it is very difficult to see the general trends of power consumption. Hence in order to filter out the high frequency noise, the readings obtained from the multimeter were averaged over a window of 100 samples (10 ms). The plot showing the power consumption of the Thinkpad during the malignant attack with the Y-axis denoting filtered power is shown in Figure 3.3. All subsequent plots of power consumption show the power as a running average of 100 samples.

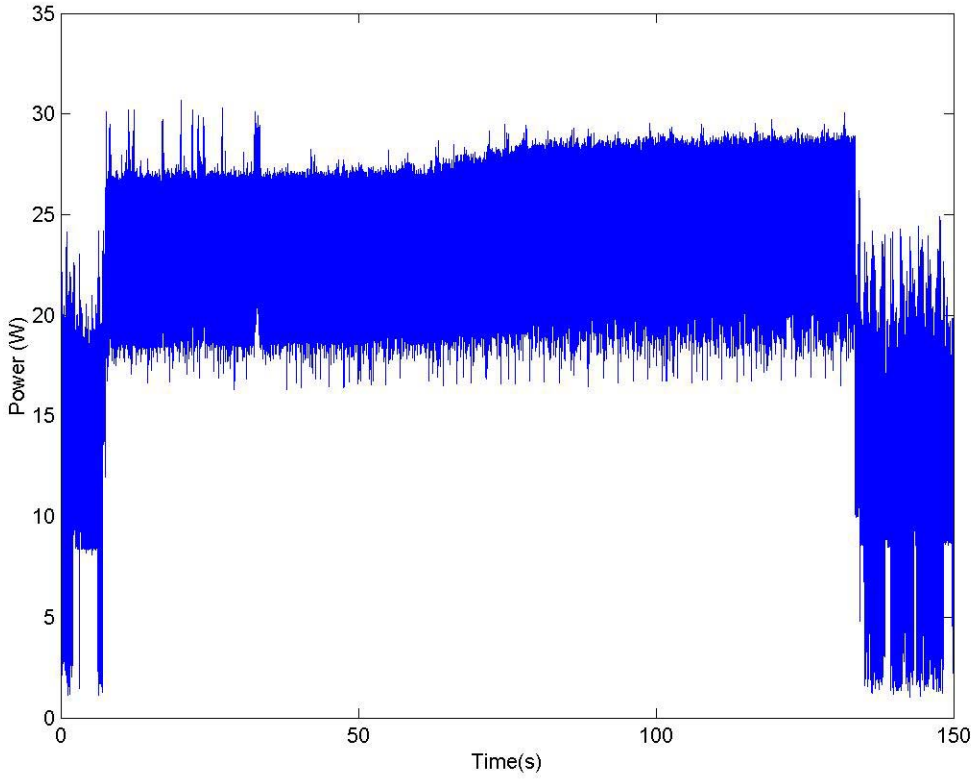

Figure 3.2: Power consumption during malignant attack on Thinkpad

3.2 Attack Implementation

This section describes the implementation of the attacks explained in Section 2.4. These attacks can be used on a variety of platforms as shown in Table 1.

Table 1: Sleep deprivation attacks implemented on different platforms

Device	Malignant Attack	Benign Attack	Service Request Attack
Thinkpad	• Cache Program • "HLT" instruction	• Animated GIF • Hash Attack • Hidden Java applet	• SSH Requests
iPAQ	• Cache Program • Restart iPAQ from suspend mode with power hungry application	• Animated GIF	• SSH Requests
Itsy	• Cache Program		

The malignant attack consisted of a program that repeatedly wrote and read an array. The length of the array was varied dynamically such that, initially, most of the array accesses resulted in cache hits, but as the array size was increased, most of the array accesses resulted in cache misses. The program kept track of the amount of time to access the entire array; when the bandwidth dropped the program could tell when the array no longer fit into the cache. A snippet of the code [27] used on the Thinkpad to implement this attack is shown in the Appendix. The function "touch_array" in the code performs basic computations on the elements of the array. As Section 3.3.1 will show, in some systems cache misses consume more power, while in other systems cache hits consume more power. Thus an attacker using this attack will likely try to tune it to the particular target system.

For the benign attack, an animated GIF was created that consisted of the same image shown repeatedly and so, to the eye, it appeared to be unanimated. To provide a comparison point, the power consumption of an unanimated version of the same image was measured. Similar to this, a webpage with a hidden, power-hungry Java applet was also implemented. Another form of benign attack was created that took advantage of

algorithmic deficiencies, as described in Section 2.3. These attacks try to implement the worst case behavior of common applications by providing malicious input data to these applications, thus increasing the power consumed by the application.

Finally, for the service request attack, repeated requests were made to an SSH server. The requests were made with a correct username but a wrong password.

All three attacks were tried on both the IBM Thinkpad and the iPAQ, but because of the network and screen limitations of the Itsy, the network service request or the benign power attack were not tried on it. On the iPAQ, a variant of the malignant attack was implemented. A program was written that waited for a fixed amount of time after the iPAQ has been suspended and then restarted the iPAQ with the screen turned off. Then, the cache program was executed causing the battery to drain its power. Such a program would have a drastic impact on the battery life of the iPAQ as will be shown in Section 4.2.

For the Itsy, only the malignant attack was tried. Due to some interesting effects of the CPU speed on power consumption, and because the Itsy was designed to allow power to be measured for both the CPU core as well as the entire system, the Itsy results for this attack are included to test the hypothesis about the CPU speed and its impact on the relative power cost of cache hits and cache misses.

3.3 Results

3.3.1 Malignant Attack

As explained earlier, the malignant attacks were implemented using a program that allocates memory of iteratively increasing array sizes and does simple computations on these arrays. Depending on the array sizes, there can either be a cache hit or miss and this causes a change in the power consumption.

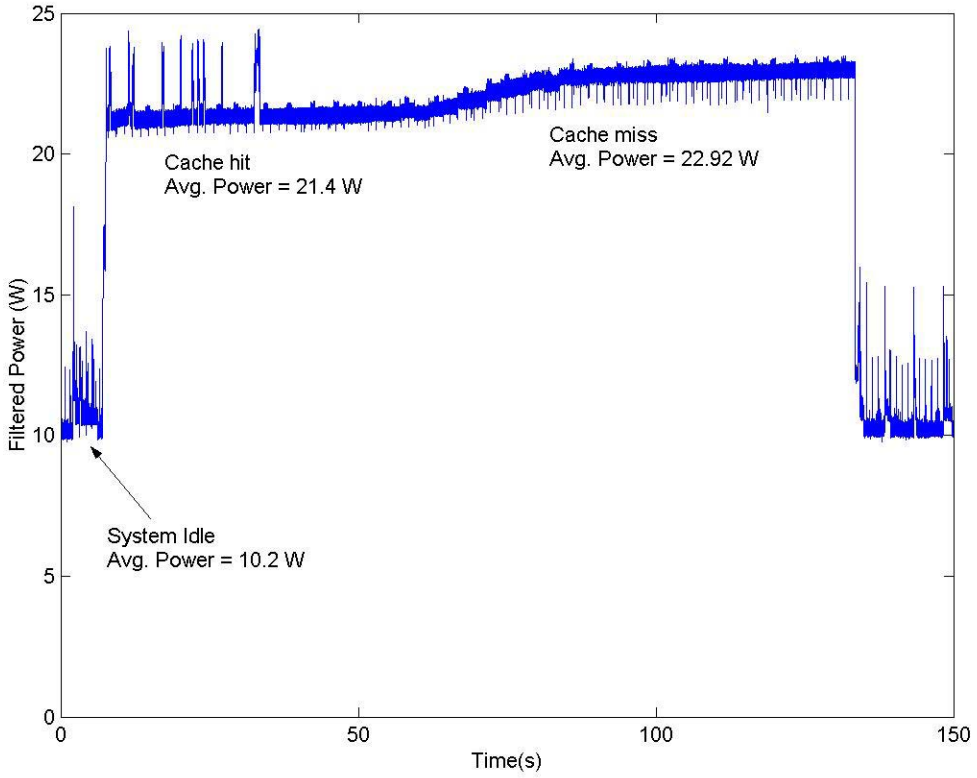

Figure 3.3: Power consumption during malignant power attack on laptop

Figures 3.3 and 3.4 show the malignant power attack on the Thinkpad and the iPAQ, respectively. The X-axis represents the time duration for which the readings were taken and the Y-axis denotes the filtered power. As shown in Figure 3.3, the power consumption of the Thinkpad almost doubled from about 10.2 W to around 21.4 W for cache hits and 22.9 W for cache misses. Thus the power consumption during a cache miss consumes around 10% more power than during a cache hit.

The iPAQ also had similar behavior to that observed on the Thinkpad with respect to an increase in power consumption when the array fit into the cache as opposed to the case when the array did not fit into the cache. The power consumed in either case was about three times the idle power of 0.42 W, as shown in Figure 3.4. During a cache hit, the iPAQ consumed about 1.10 W while during a cache miss it consumed 1.17 W.

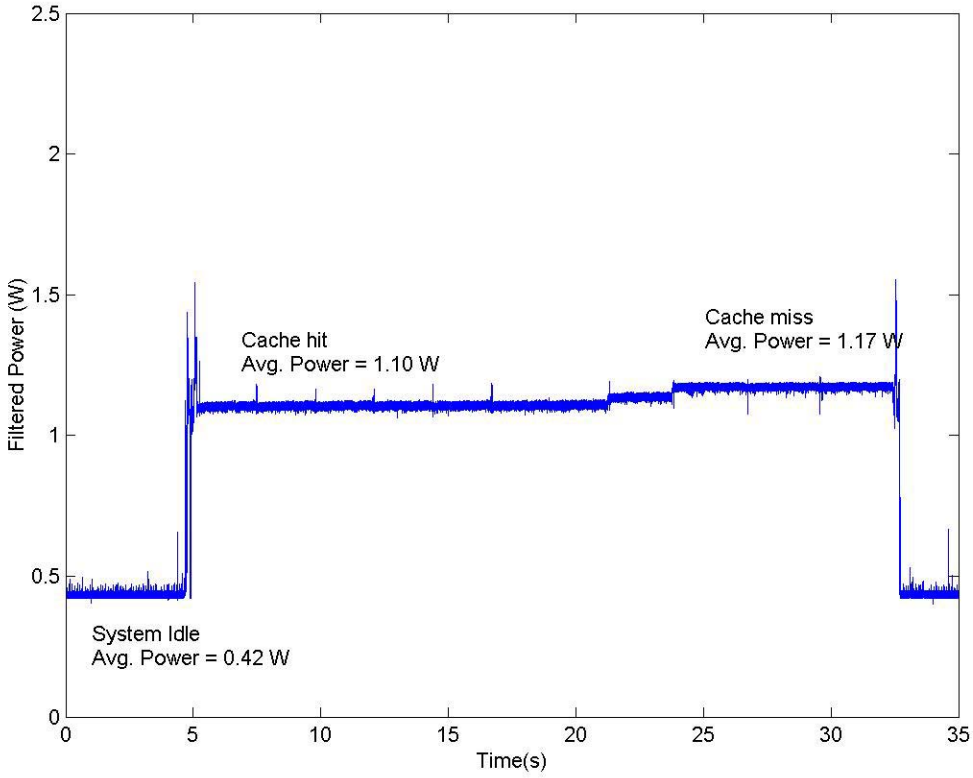

Figure 3.4: Power consumption during a malignant attack on iPAQ

For the Itsy, the results are even more interesting as shown in Figure 3.5 and Figure 3.6. The Itsy Research prototype has the option to allow the user to change the operating CPU frequency. The same cache program was run on the Itsy, with two different processor speeds; 206 MHz and 59 MHz. When run at 206 MHz, the Itsy consumed 10% more power in a cache hit than a cache miss, while at 59 MHz, the Itsy consumed 10% more power in a cache miss than a cache hit.

A table indicating the idle power and the power consumed during cache hits and cache misses is shown in Table 2. The last column indicates the percent increase in power consumption between the device being idle and the device being under attack.

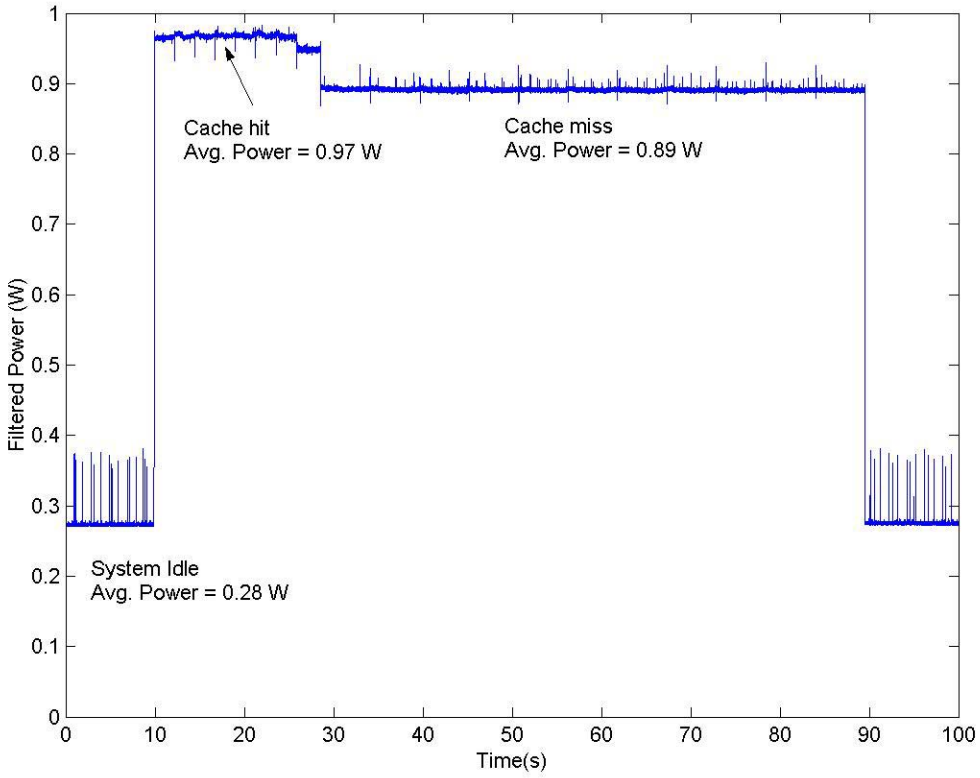

Figure 3.5: Power consumption during malignant power attack on Itsy at 206 MHz

Table 2: Power consumption of the devices under malignant attack

	Average Idle Power (W)	Average Power when under attack (W)		Percent increase in power (%)
		Cache hit	Cache miss	
Thinkpad	10.2	21.40	22.92	125
Itsy @ 206 MHz	0.28	0.97	0.89	350
Itsy @ 59 MHz	0.22	0.42	0.50	127
iPAQ	0.42	1.10	1.17	149

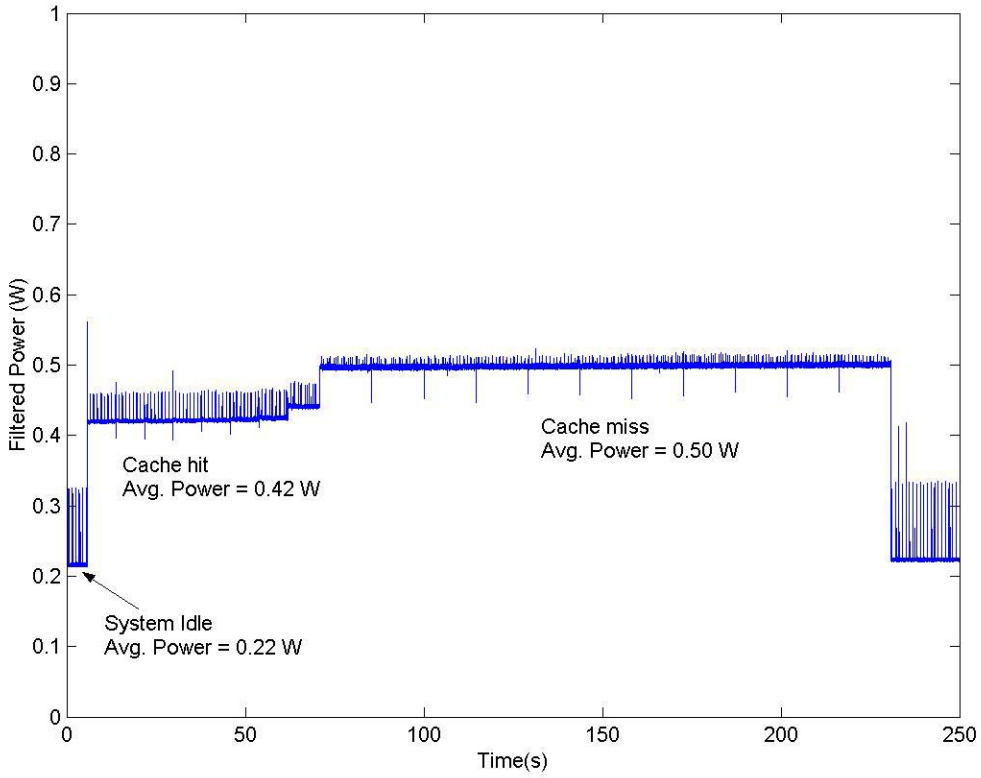

Figure 3.6: Power consumption during malignant power attack on Itsy at 59 MHz

These plots indicate that if an attacker wants to drain the battery of a device by exploiting the cache behavior of the device under attack, he must know the hardware configuration of the device. In particular, the attacker would be required to know the frequency at which the device under attack is operating and also whether at that frequency, a cache hit or cache miss will consume more power. One option for the attacker is to maintain an exhaustive database of all possible devices and their clock frequencies. An easier and more practical solution would be to get the required information from the battery state.

The Thinkpad's Advanced Configuration and Power Interface (ACPI) [2] was used to implement a program that tuned itself to maximize power consumption based on the cache behavior. This interface provides user-level access to files that store information pertaining to the battery state. The pseudo-code for this tuning program is as follows:

- Run the cache program for an iteratively increasing array size.

- Note down the time taken to access each array. Using this time calculate the bandwidth.
- Find out the maximum (MAX_BAND) and minimum (MIN_BAND) bandwidth and the corresponding array sizes, MAX_ARRAY and MIN_ARRAY respectively.
- Run the cache program again, this time only for MAX_ARRAY array size.
- Store the remaining battery capacity at the beginning and the end of the program. Repeat the same procedure for MIN_ARRAY array size.
- Compare the power consumed while running the cache program for MAX_ARRAY and MIN_ARRAY size.

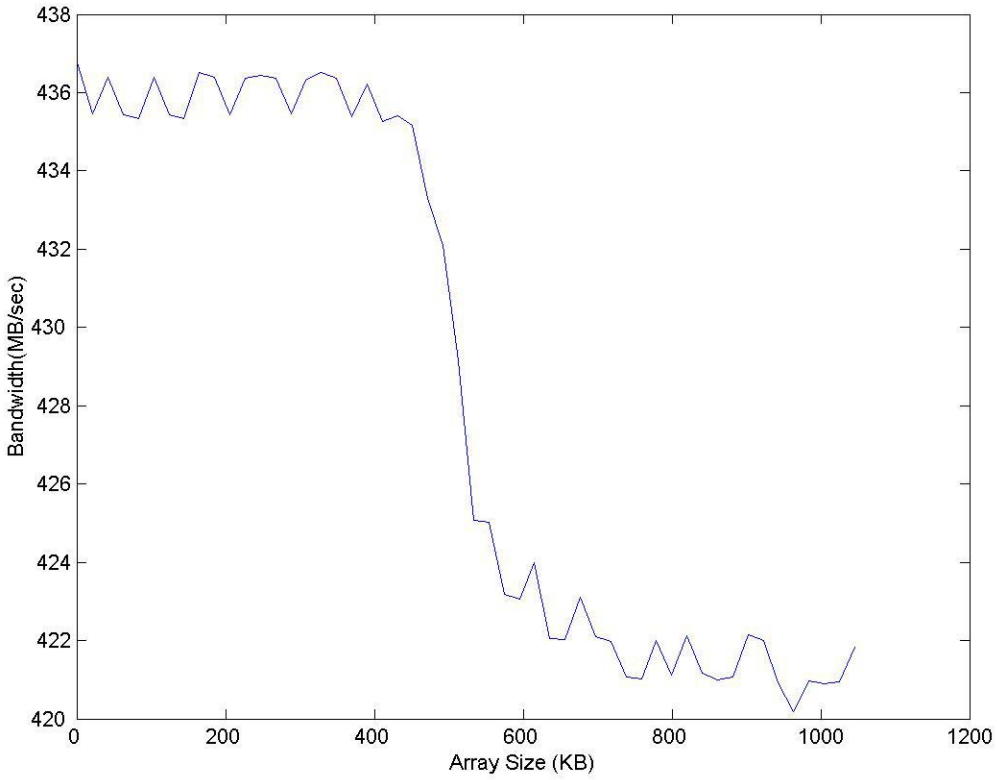

Figure 3.7: Bandwidth of the array accesses on Thinkpad during cache program execution

The array size that causes a greater reduction in the capacity of the battery decides whether a cache hit or a cache miss caused greater power consumption.

The relation of the bandwidth and array size is shown in Figure 3.7. The Thinkpad has a Pentium III processor with a built in 512 KB cache. As shown in Figure 3.7, the bandwidth of array accesses reduces as the array size increases and goes beyond around 500 KB. Using this plot we can get two values of array sizes for maximum and minimum bandwidth that would cause a cache hit and a cache miss, respectively.

While an attack that makes the device leave the idle state can increase the power consumption drastically, the power consumption of the device under attack can be increased even more drastically if an attacker can cause the device to execute power hungry applications when the device is actually supposed to be in low power sleep or suspend mode. In order to implement this attack the properties of an iPAQ in suspend mode were exploited. An iPAQ enters suspend mode when the user presses the power

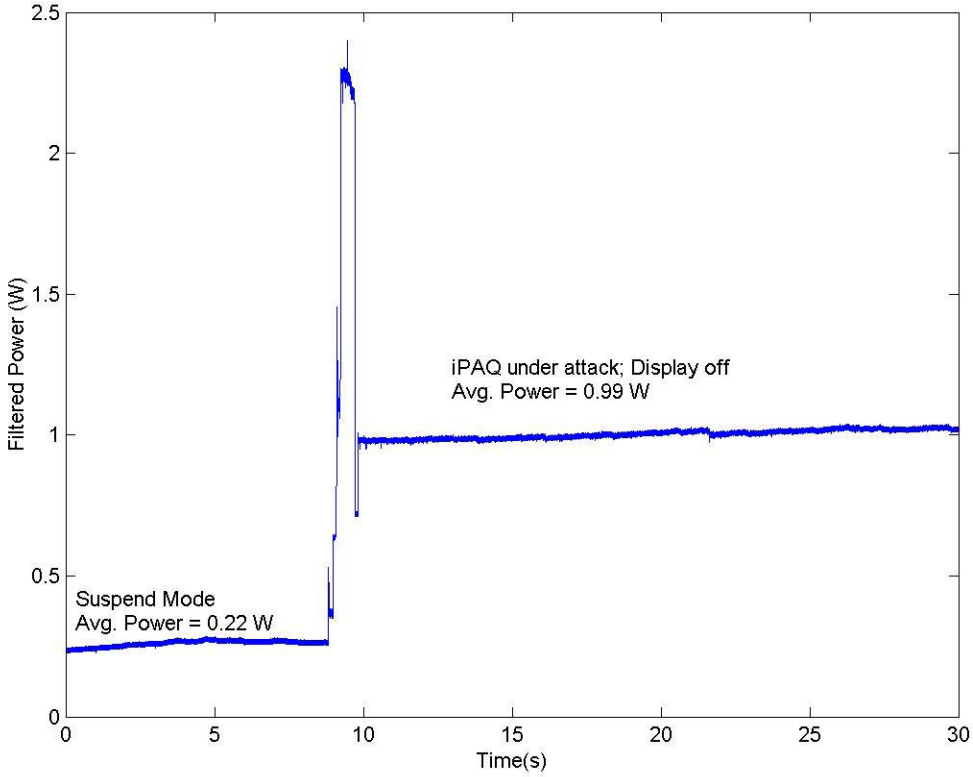

Figure 3.8: iPAQ leaving suspend mode and executing power hungry application with display turned off

button. In this state everything except the timer and the RAM is turned off. The timer was used to trigger an event causing the iPAQ to leave suspend mode. The display was turned off as soon as the iPAQ left suspend state to escape notice of the user. Once the iPAQ was turned on, a power hungry application was executed.

Figure 3.8 shows the power consumption of the iPAQ during the attack implementation. In suspend mode the iPAQ consumes 0.22 W and this increases four times to around 1 W when the attack is implemented. Such an attack is truly devastating to the user because it could potentially reduce the battery life of the device by around four times. In fact if the power management features of the iPAQ are improved so that the power consumed while in suspend mode is reduced, the impact of this attack would be even more. This will be explored further in Chapter 4.

Another sophisticated, yet effective way of draining the battery would be to modify the idle loop of the operating system of the device under attack. Although, this attack was not implemented by actually changing the binary code of the operating system on the DUA, a workaround was used to show the effect this could have on the battery life. This workaround involved appending a line to the boot loader of the linux OS on the Thinkpad. This change notified the OS not to execute the "HLT" instruction in the idle loop the next time it booted up. It would instead enter into a busy loop when idle, defeating a major purpose of the power management and thus reducing the battery life significantly. Figure 3.9 shows the difference in idle power of the two cases on the Thinkpad with Linux loaded on it. Another point to note is the idle power of the laptop is around 2 W more than previously shown values. This is because the test was run on the same laptop, but with Redhat Linux 7.2 loaded on it instead of Windows 2000. The difference in the power management features of the two operating systems results in different idle powers for the same hardware.

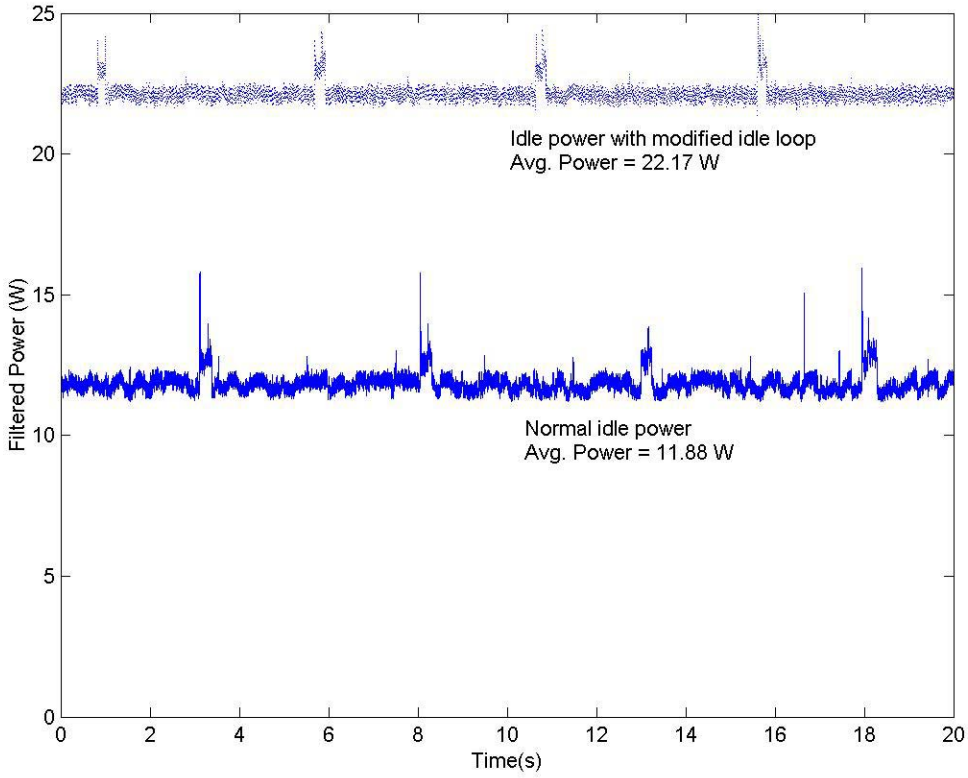

Figure 3.9: Idle power consumption of Thinkpad with normal and busy idle loop

3.3.2 Benign Power Attack

To implement this attack, an animated GIF was created with only one frame repeated at a very high frequency. Thus to the user, this GIF would seem unanimated although it would be draining the battery power significantly as shown in Figure 3.10. This figure shows the difference in power consumption for the Thinkpad while viewing an animated GIF of a single image and a non-animated GIF of the same image. The power consumption increases from 10.2 W to 18.4 W, an increase of about 80%.

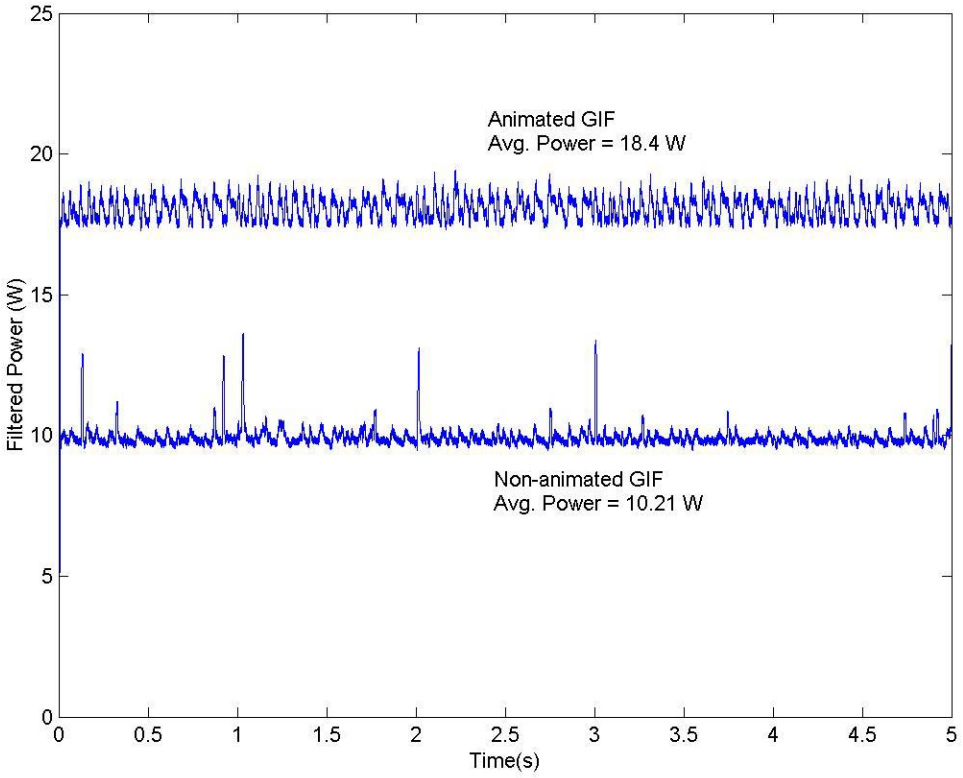

Figure 3.10: Comparison of power consumption of animated and non-animated GIF on Thinkpad

The same GIFs when viewed on the iPAQ caused the power consumption to increase by nearly a factor of three, from around 0.42 W to about 1.15 W. These results are shown in Figure 3.11.

Table 2 shows the values of power consumption for the iPAQ and Thinkpad in idle state and also when viewing an animated and unanimated GIF. The last column indicates the percent increase in power consumption when viewing an animated GIF compared to the idle power.

Table 3 : Power consumption of the devices under benign attack

	Average Idle Power (W)	Average Power when under attack (W)		Percent increase in power (%)
		Non-animated GIF	Animated GIF	
Thinkpad	10.2	10.21	18.4	80
iPAQ	0.42	0.42	1.15	188

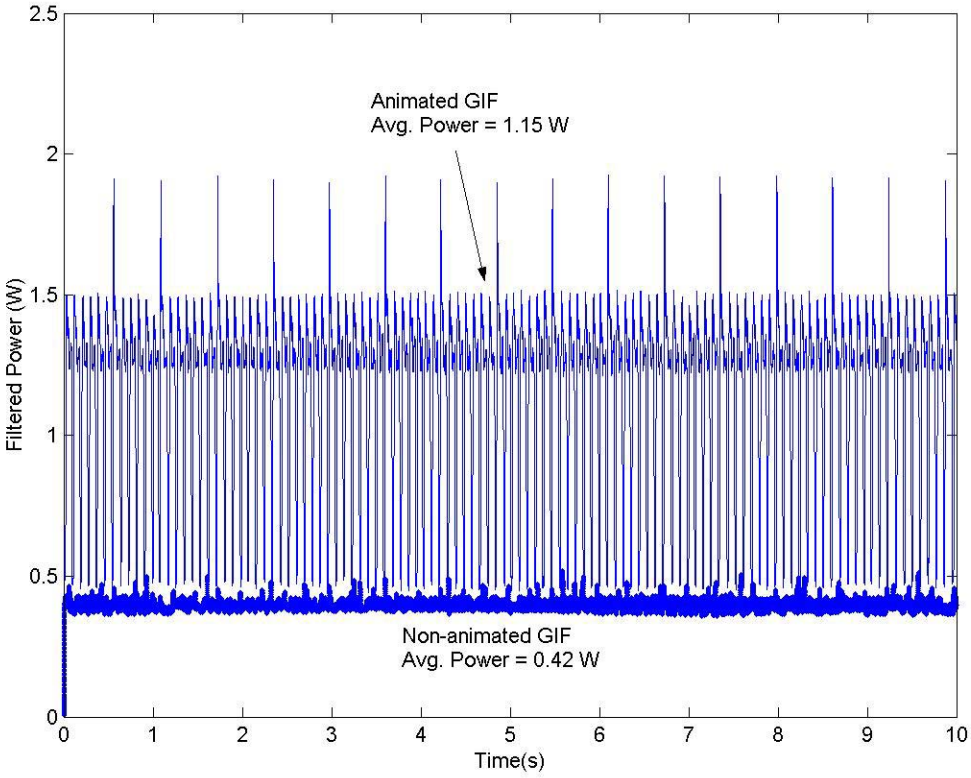

Figure 3.11: Comparison of power consumption of animated and non-animated GIF on iPAQ

Similar to the animated GIF, a webpage could be modified to run a Java applet in the background without the user taking notice of it. Any browser with Java support would be able to run the code without the user knowing it. This was implemented on the Thinkpad and the results are shown in Figure 3.12. The Java applet starts running in the background as soon as the user launches the webpage. This particular Java applet is a variant of the cache program explained in section 3.3.1. As shown in the figure the Java applet causes the power consumption to increase from around 10 W to 17 W (an increase of 70%).

These attacks could be easily implemented using an image embedded in a webpage. An attacker can modify the webpage to include an animated GIF that the user interprets to be unanimated or can include a power hungry Java applet invisible to the user. This would cause the device under attack to significantly increase its power consumption and thus draining its battery resources.

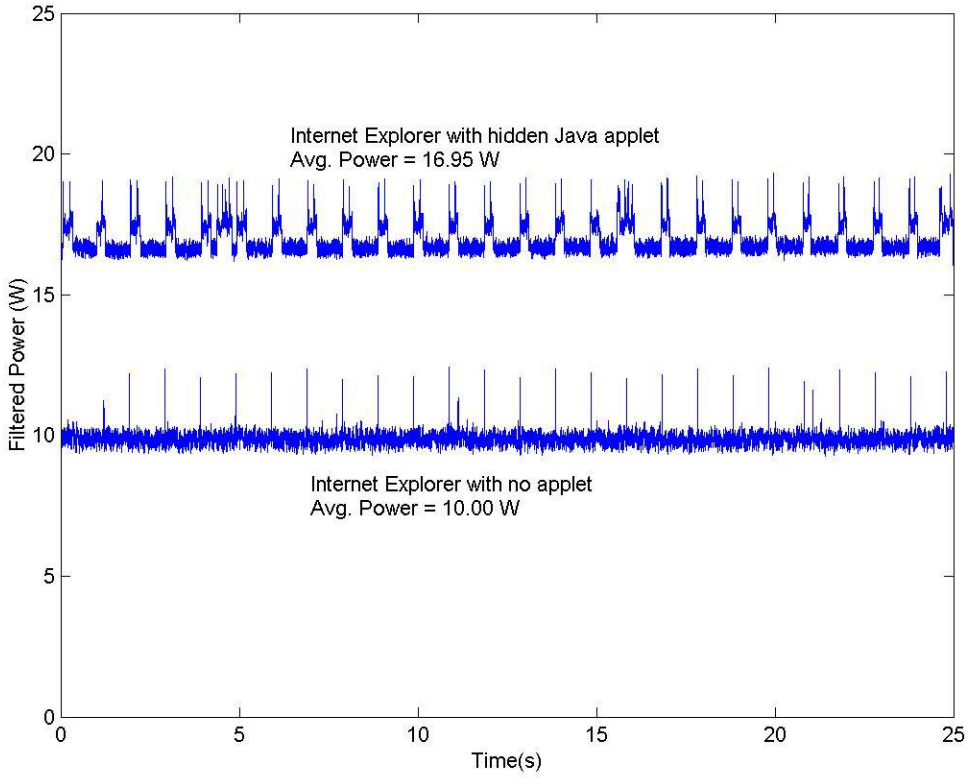

Figure 3.12: Power consumption of Thinkpad while running IE with and without a hidden applet

The principle of modifying the input to a program to make it execute its worst case behavior was used in Algorithmic Complexity Attacks [1], as described in Section 2.3. These attacks take advantage of algorithmic deficiencies in many common applications to increase the power consumption of the DUA. The attack shown in Figure 3.13 is implemented by exploiting vulnerabilities in Perl's hash table algorithm.

This feature has been used while implementing the hash attack on Perl 5.6.1. A program was created that took input from a file containing 10,000 strings of malicious data and inserts this data into an associative array. Executing this program caused the power consumption of the Thinkpad to increase from around 12W to about 20W (an increase of 75%). Here again the idle power is slightly greater than previously shown values because the test was run with Redhat Linux 7.2 loaded on the Thinkpad.

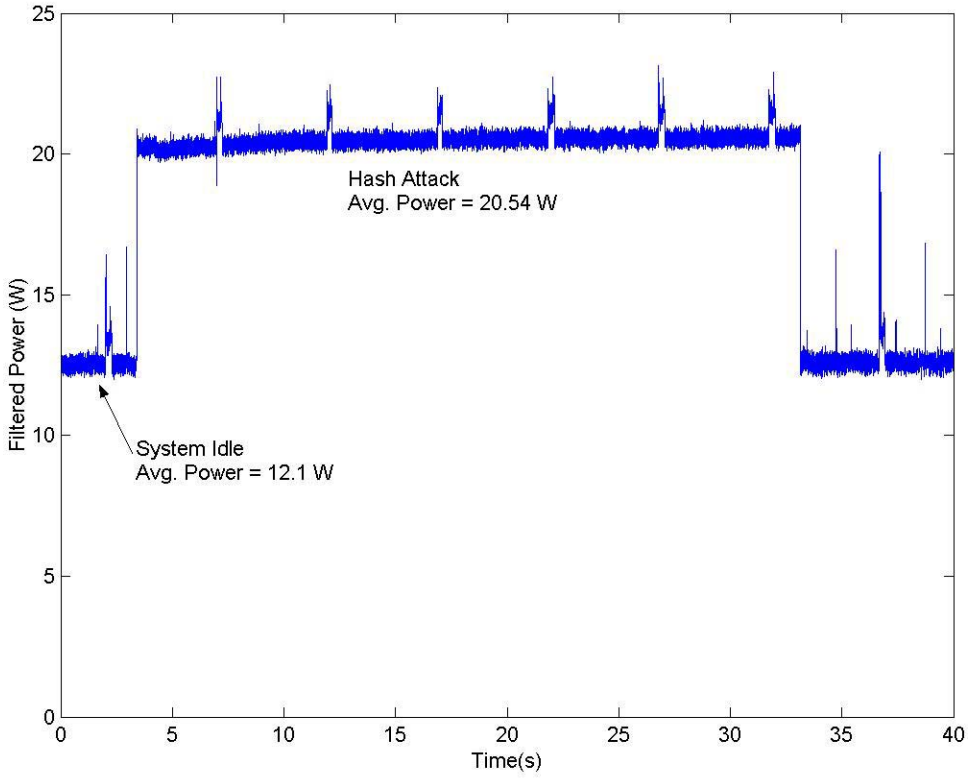

Figure 3.13: Power consumption of Thinkpad during a Hash attack

3.3.3 Service Request Attack

Figure 3.14 shows the power consumption of the laptop running an SSH server while under a service request attack. When the service request attack began the power consumption increased by around 15%. A similar attack was implemented on the iPAQ running an SSH server shown in Figure 3.15. For implementing this attack, an iPAQ with the Linux operating system was used. The idle power of the iPAQ running Linux is around 2W and the service request attack causes an increase of about 40% in the power consumption. The increase in the idle power of the iPAQ is because of the wireless card present in the expansion slot.

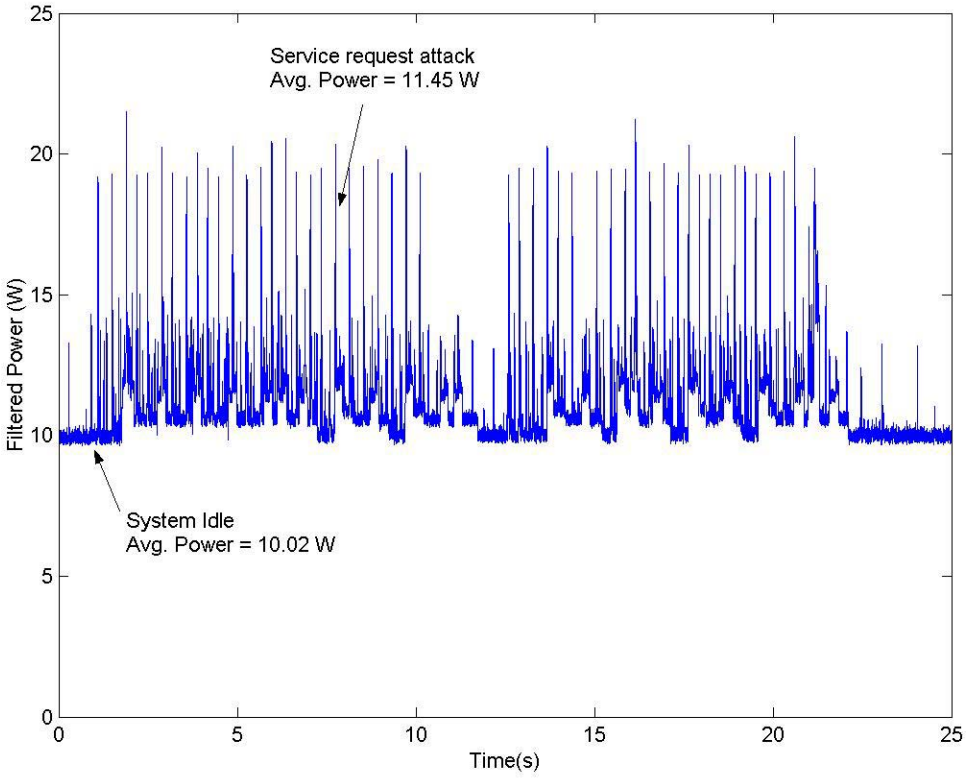

Figure 3.14: SSH requests made to Thinkpad with wrong password

The power profile of the iPAQ shown in Figure 3.15 is interesting since it clearly shows the effect that establishing an SSH connection and the resulting computation for verifying the password have on the power consumption of the iPAQ. The script written for implementing this attack opens an SSH connection to the iPAQ five times and each connection asks for the password three times. Figure 3.15 also shows five peaks in the power consumption at regular intervals once the service request attacks begin; corresponding to the five requests to open SSH connections. These five peaks can be attributed to the key information exchange that takes place when an SSH client connects to a server so that the two systems can correctly construct the transport layer [3]. The following steps occur during this exchange:

- Key exchange
- Negotiate the public key algorithm to be used
- Negotiate the encryption algorithm to be used

- Negotiate the message authentication algorithm to be used
- Negotiate the hash algorithm to be used

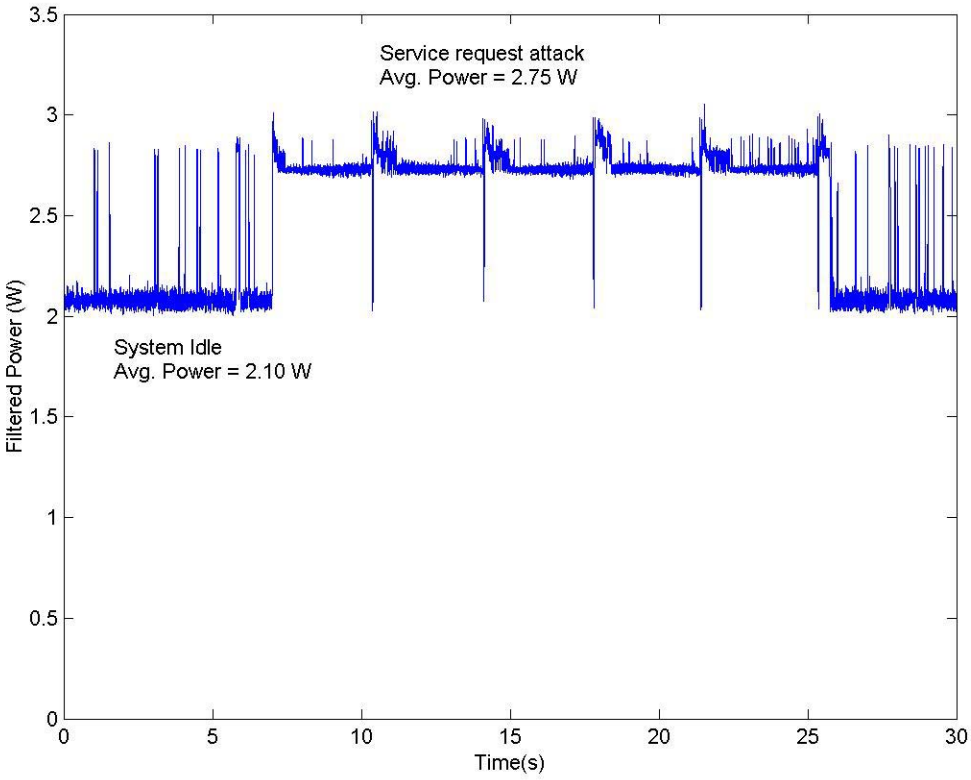

Figure 3.15: SSH requests made to the iPAQ with wrong password

Once these values are decided the client authenticates itself using either a password or a key-encoded signature. Since password is the most common case, it is also used here. The decoding algorithm used for authenticating the password is responsible for the power consumption to remain more than the idle power.

This chapter has described the implementation of the three types of attacks and the impact on the power consumption. The next chapter describes a battery usage model that provides an estimate of the battery life depending on the time spent by the device in one of the possible sleep, idle and active states.

Chapter 4

4 Battery Usage Model

The previous chapter explained the different types of attacks implemented on a variety of platforms and the corresponding results. This chapter presents a model for battery usage based on one presented in [12]. The goal of this model is to provide an estimate of the battery life depending on the time spent by the device in one of the possible sleep, idle and active power management states. The model will then be used with the results presented in the previous chapter to see the effect that these attacks can have on the battery lifetime of the device under attack.

4.1 Derivation

A device typically enters into active state when performing some actions for the user like displaying something on the screen or performing some computations. In the remaining times, the device enters into low power sleep or standby mode to conserve battery power. Taking this into consideration, the following metrics are defined for device power consumption [12]:

P_{avg_active} : The average power consumed by the device during active mode

P_{sleep} : The power consumed by the device during sleep mode

D : Usage Duty Factor, the fraction of time device is on

PFR : Power Factor Ratio $\left(\frac{P_{avg\_active}}{P_{sleep}} \right)$

Considering these metrics, the average power consumed by the device can be expressed as:

$$Average\_Power = P_{sleep}(1 - D) + P_{avg\_active} D$$

Including the Power Factor Ratio (PFR) into the expression, the power consumed would be:

$$Average\_Power = P_{sleep}(1 - D + PFR * D)$$

Assuming that the battery life is inversely proportional to the average power, the battery life could then be expressed as:

$$Battery\_Life = \frac{Battery\_Capacity}{Average\_Power}$$

From these expressions, it can be inferred that in order to minimize battery life, one should increase the Duty Factor (D), Power Factor Ratio (PFR) or both. Typically, the values of D are reported to be very small. Kamijo et. al. report a value of about 0.0035 for Palm Pilots, which amounts to a usage of around 5 minutes a day [12]. Since the values of D are typically very small, it would be easier for an attacker to increase the power consumption of the device. This can be achieved by causing the device to consume much more power in the active mode than it would under normal circumstances. Hence the battery life would be directly related to the value of the PFR. The higher the PFR, the lower is the battery life and vice-versa. Given the range of values for PFR

in [12], an attacker could reduce the battery life of mobile computers by a factor of 30 to 280.

This particular model takes into account the average power consumed by the device while in active mode. However, in practical usage the power consumption of a device when it is active varies depending on the activity being performed. Under normal circumstances, when the device is not performing any activity, the power consumption would be low because the system is idle. However, the system can be brought into the active state leading to an increase in power consumption because of many factors like a hard drive spinning up, the CPU being used to perform some computations or wireless communication activity. Hence in order to give this model a more practical dimension, a few more parameters are incorporated into this model:

P_{active} : The power consumed by the device during active mode

P_{idle} : The power consumed by the device when idle

A : Active Duty Factor, time spent in active mode compared to time spent in idle mode

The value of P_{avg_active} would then be replaced by:

$$P_{avg\_active} = P_{idle}(1 - A) + P_{active}A$$

Incorporating these metrics into the expression for $Average\_Power$, the expression for power consumed would become:

$$Average\_Power = P_{sleep}(1 - D) + D[P_{idle}(1 - A) + P_{active}A]$$

This expression will give a better estimate of the battery life because it takes into account the maximum possible power consumption of the device and the time the device spends consuming that much power. An attacker would try to make the device spend

most of its time in the active mode (A = 1) whenever the device is not sleeping and prevent it from going into the lower power idle mode.

The next section makes use of this model to show the effect of a sleep deprivation attack on the battery life of the device under attack.

4.2 Results

The following table shows the power consumption values for the iPAQ and Thinkpad when in sleep, idle and active modes. Table 4 shows the values of power consumption for the iPAQ and the Thinkpad based on the measurements from the previous chapter.

Table 4: Power consumption of iPAQ and Thinkpad in sleep, idle and active mode

	iPAQ		Thinkpad (W)
	Without Wireless Card (W)	With Wireless Card (W)	
P_{sleep}	0.2	0.2	0.23
P_{idle}	0.42	2.1	10.2
P_{active}	1.17	2.75	22.9

Using the values of the iPAQ without the wireless card the relation between battery life and usage duty factor for different values of Active duty factor can be plotted, as shown in Figure 4.1. The battery life is normalized to the battery life when the device is in sleep mode all the time. The duty factor is shown on a logarithmic scale.

Typically, an iPAQ would be used only about 5 minutes/day [12] which corresponds to a very low duty factor. As shown in Figure 4.1, the battery life of the iPAQ is dominated by the power consumption in sleep mode at lower duty factors. Also at low duty factors, since the power consumption in sleep mode is not an order of magnitude lower than the idle or active mode, the effect of active duty factor (A) on the battery life is negligible. This is the reason that at low duty factors, the battery life of the iPAQ would not be affected a great deal when under a sleep deprivation attack.

In order to reduce the battery life of the iPAQ an attacker would have to increase the duty factor (D). This can be achieved by implementing an attack that causes the iPAQ to wake up from suspend state and execute a power hungry application. Such an attack could potentially reduce the battery life by a factor of five because the operating point would move from a low duty factor on the graph to an operating point with $D = 1$ and $A = 1$.

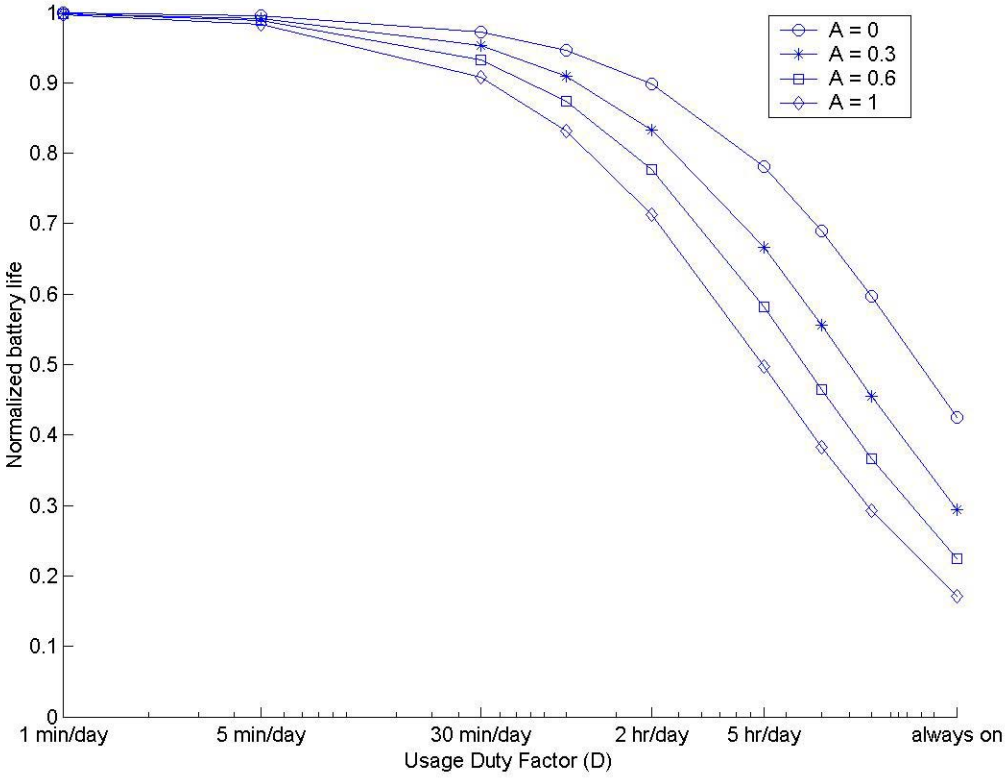

Figure 4.1: Battery life verses Usage Duty Factor (D) for iPAQ

Figure 4.2 shows the plot using the values from the laptop. The power consumption of the laptop during the sleep mode is much less compared to the idle and active mode power consumption. Hence the effect of the sleep power on the battery life is almost negligible. In the case of a laptop, typically the usage duty factor would be around 3 hours/day [21]. For these typical values of usage duty factor (D) of the laptop, the battery life would be also related to the active duty factor (A). This would result in the

difference in the power consumption of the idle and active state being reflected in the battery life of the device. In this case, the active power is almost twice the value of the idle power. As a result, the battery life reduces approximately by half when the laptop goes from an always idle state $(A = 0)$ to an always active state $(A = 1)$.

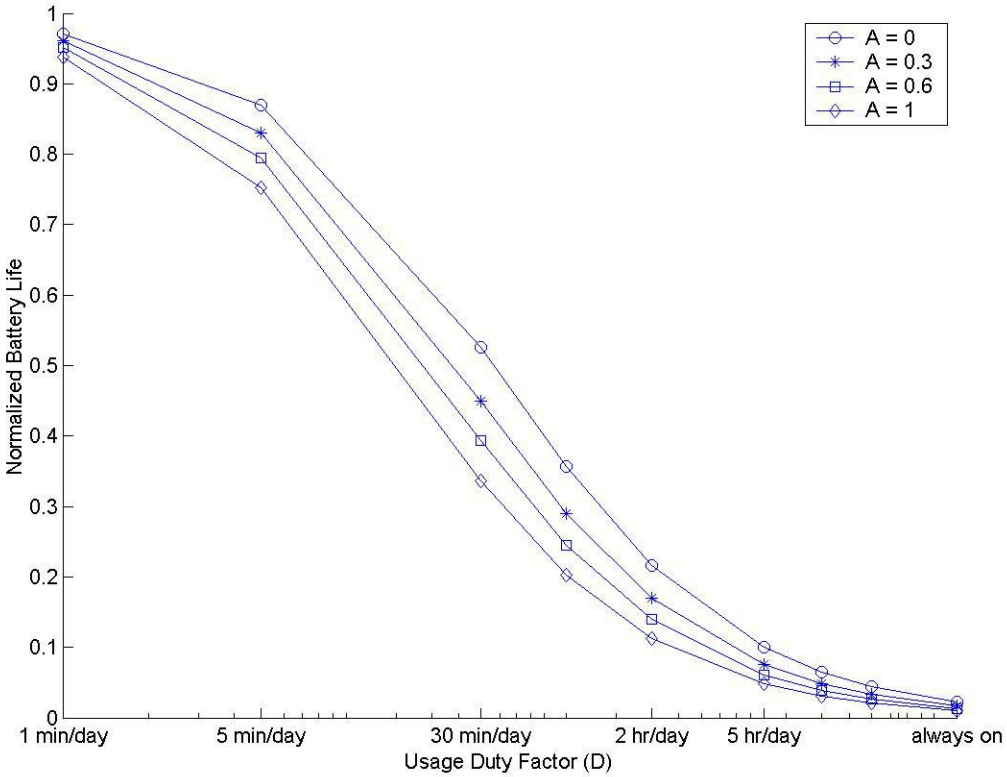

Figure 4.2: Battery Life versus Usage Duty Factor (D) on laptop

This model can be applied to a wide variety of systems with different values of sleep, idle and active powers. Based on the type of system power consumption values, estimations can be made for the battery life under different conditions. For example, this model was used to gauge the effect of a denial of service attack on a device whose idle power is an order of magnitude greater than the sleep power but the active power only slightly more than the idle power. The resulting plot is shown in Figure 4.3.

As shown in Figure 4.3, the battery life is greatly affected by an increase in the usage duty factor (D). A slight increase in the duty factor would cause the battery life to reduce

drastically. Also, in this case since there is not much of a difference between the active mode and idle mode power consumption, the effect of active duty factor (A) on the battery life is negligible. For such a system, an attacker trying to cause a denial of service attack would serve his purpose by simply trying to make $D = 1$, irrespective of the value of the active duty factor (A).

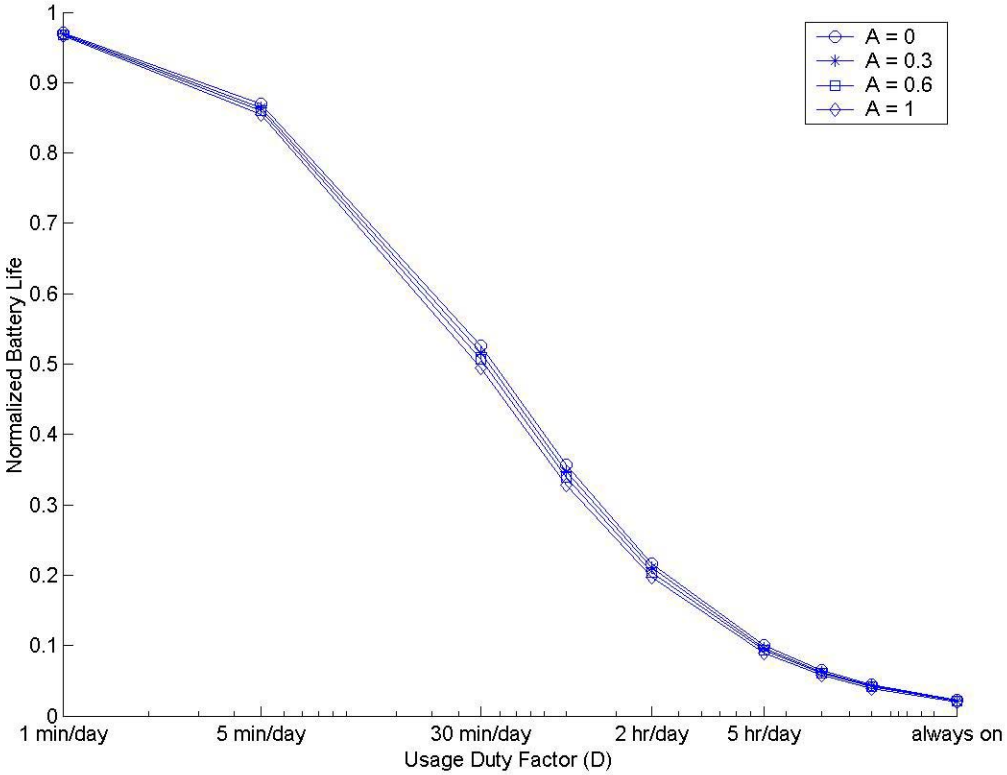

Figure 4.3: Battery Life versus Usage Duty Factor (D) for a system with idle power >> sleep power

This model was also applied to another system with the property that the power consumption in active mode is much greater than that during idle mode and the idle mode power consumption is only slightly greater than the sleep power. The resulting plot is shown in Figure 4.4.

As shown in Figure 4.4, the battery life is greatly dependent on both the usage duty factor (D) and the active duty factor (A). For example, a usage duty factor of around 2 hr/day would cause the battery life of the device to be reduced by half when the device

goes from an always idle state to an always active state whereas an always on state $(D = 1)$ would cause the battery life to be reduced by a factor of seven for the same transition. For such a system the attacker would try to make the active duty factor as high as possible $(A = 1)$ and if possible, the attacker would also try to keep the device in an always on state $(D = 1)$.

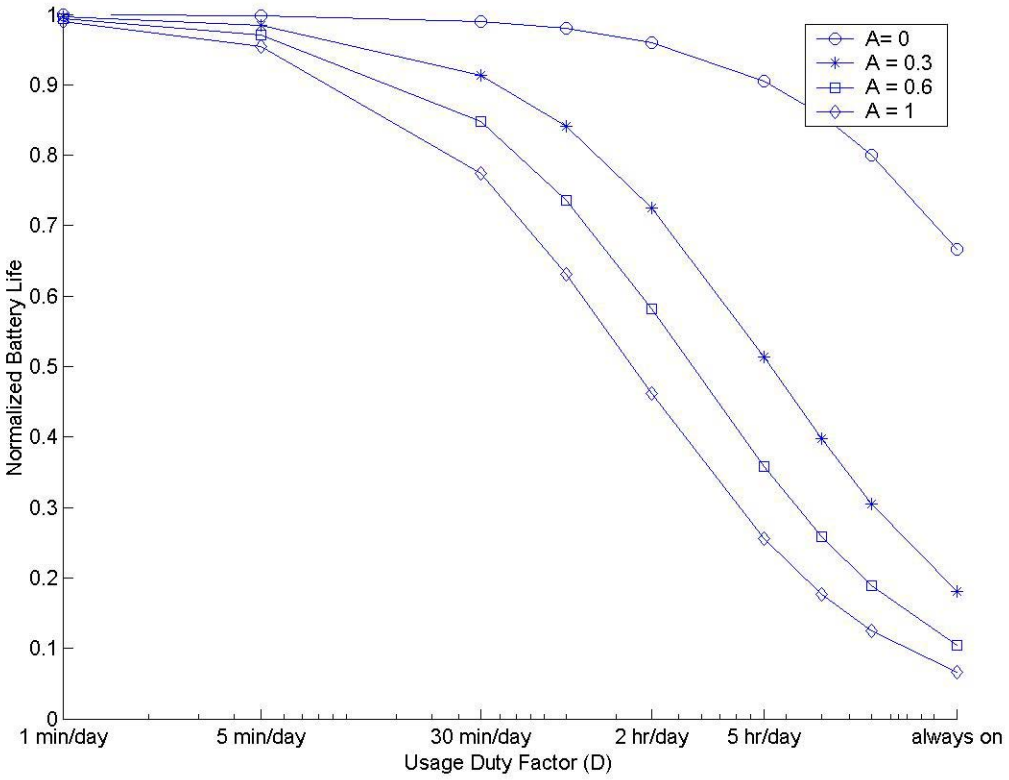

Figure 4.4: Battery Life versus Usage Duty Factor (D) for a system with active power >> idle power

4.3 Summary

The model presented in this chapter tries to incorporate all possible metrics involved in estimating the battery life of devices. Any device, whose power consumption during the sleep, idle and active power states is known, can have its battery life estimated based on the typical times spent in each state.

The model provides an estimate of the effect of a sleep deprivation attack on a device's battery life for typical usage duty factor values. For example, a laptop under a malignant attack with a typical usage duty factor could have its battery life reduced by a factor of two. Also, if systems are provided with a functionality to be able to come out of suspend mode either remotely or through a program, an attacker could exploit this feature to shift the operating point of the graph to the right towards an always on always active state. This would cut down on the battery life drastically. An attack on an iPAQ that causes it to come out of suspend mode and execute power hungry applications could potentially reduce the battery life by a factor of five.

The model applied to the hypothetical systems shows that the characteristics of the system determine whether or not an attacker has to try to go towards a state with $A = 1$, $D = 1$ or $A = 0, D = 1$ in order to be able to maximally drain the battery. An always on always active state is always going to have the maximum effect but in some systems, like that shown in Figure 4.3, an attacker could also have the desired effect with $A = 0$, $D = 1$ state because of the negligible difference between the idle and active powers.

Such a model would also help developers in designing systems with an eye towards possible sleep deprivation attacks and the impact of these attacks on the battery life.

Chapter 5

5 Conclusion

This thesis has explored DoS attacks on batteries of mobile devices. These power related attacks render a device inoperable by draining the battery long before the expected lifetime.

The methods that could be adopted by an attacker to implement these attacks fall into three categories:

- Malignant attack: application code or kernel binary is modified to increase power consumption
- Benign attack: the device is made to execute a valid but energy-hungry application
- Service Request attack: repeated requests are made to the device over the network

These attacks were implemented on palmtops and a laptop to study their effect on the power consumption of the device. These are the first real examples of DoS attacks on mobile devices. The results show that the power consumption of the devices could potentially increase by a factor of 2-3 in some cases. With time, these attacks are expected to become more potent and sophisticated.

Finally, a model was proposed to help estimate the battery life of the mobile device based on the system parameters and typical usage characteristics. This model can be used to gauge the effect of a sleep deprivation attack on the battery life of the device. For example, a malignant attack on a laptop could reduce its battery life by a factor of two. An attack on the iPAQ that forces it out of suspend mode could potentially reduce the battery life by a factor of five.

This model shows how an attacker would try to maximize the battery drain based on the power consumption characteristics of the system. This is illustrated by the results of the model shown for the two hypothetical systems. An attacker would always try to move towards an always on always active state but in some systems, an attacker can also get the desired effect by simply keeping the device on because of the negligible difference between the idle and active powers.

5.1 Future Work

DoS attacks on batteries have not gained much attention in the literature yet. Research in this field is still in its rudimentary stage. The attacks presented in this thesis are meant to raise awareness among mobile system developers about this attack. Research into new types of attacks on batteries would help developers in incorporating the necessary framework for detecting vulnerabilities to these attacks at design time.

The next step in this research would be to try to devise methodologies to detect these attacks. A proposed power secure architecture that would detect these attacks by guaranteeing a certain minimum battery life would be a first step in this direction [30]. Multi-layer authentication and energy signature monitors are two major components of this architecture. Multilayer authentication is used to guard against service request attacks. Additional resources are committed to only those requesters which have gained a certain level of trust. The energy signature monitor is designed to detect intrusions that get past the authentication scheme and are using up battery power by executing power hungry applications. Although research in this field is underway, future work involves implementing and testing this architecture on a variety of platforms.

Appendix A: Code for Cache program

This code is based upon cachsize.c, by Phil Koopman [27].

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define K  * 1024
#define M  * 1024 * 1024
/* Tune TUNE for machine speed;  larger number runs slower */
#define TUNE  10 M
#define MAXSIZE ( 1 M)


int touch_array(int *array, int num_ints, int iterations)
{
      int i;
      int sum=0;
      int *p, *limit;

      limit = &(array[num_ints-1]);

      for (i = 0; i < iterations; i++)
      {
            /* touch elements; pointer match results in faster code */
            for (p = array; p != limit; p++)
            {
                  sum += *p;
            }
      }
      return(sum);
}


void main(void)
{
      int *a;           /* data array */
      int size, j, iter;
```

```
	int num_ints;	/* number of ints in the array */
	int result;
	double x, elapsed, rate, tot_elapsed = 0;
	clock_t start, finish;

	printf("\n\nTesting tuning value of %d\n", TUNE);
	printf("Array size, Iterations\n");

	for (size = 1 K; size < MAXSIZE; size += 20 K )
	{
		a = (int *) malloc(size);
		num_ints = size / sizeof(int);

		/* initialize array to a constant value */
		for (j = 0; j < num_ints; j++)  { a[j] = 1; }

		/* scale iterations with array size, so run time is about
constant */
		x = size;
		x = (TUNE) / x;
		iter = (int) (x * 100);

		/* pre-load cache to avoid compulsory misses */
		result = touch_array(a, num_ints, 1);

		/* start timer */
		printf("%9d, %10d\t ", size, iter);
		start = clock();

		/* touch the array repeatedly */
		result += touch_array(a, num_ints, iter);

		/* stop timer */
		finish = clock();

		elapsed = (double)(finish - start) / CLOCKS_PER_SEC;

		rate = size;  rate = rate * iter / elapsed;

		tot_elapsed +=elapsed;

		if (result)
			printf("%.3f, %.f, %d\n", elapsed, rate, num_ints);

		free(a);
	}  /* next bigger array size */
}
```

Bibliography

[1] S. A. Crosby and D. S. Wallach, “Denial of Service via Algorithmic Complexity Attacks” in *Usenix Security Symposium*, pp. 29-44, August 2003

[2] Intel Corporation, ACPI web site, *http://developer.intel.com/technology/iapc/acpi/*

[3] Redhat Inc., Redhat Linux 7.3: The Official Red Hat Linux Reference Guide, Chapter 10: The SSH Protocol, *http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/ref-guide/s1-ssh-protocol.html*

[4] J. Flinn and M. Satyanarayanan, “PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications”, in *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 2-10, February 1999.

[5] M. Viredaz, *The Itsy Pocket Computer Version 1.5 User's Manual*, Compaq Western Research Laboratory Technical Note TN-54.

[6] A. D Wood and J. A. Stankovic, “Denial of Service in Sensor Networks”, in *Computer, Volume: 35, Issue: 10*, pp.54 – 62, October 2002.

[7] 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.

[8] T. Aura, P. Nikander, and J. Leiwo, "DOS-Resistant Authentication with Client Puzzles," in *Proc. Security Protocols Workshop*, pp. 170-177, Springer--Verlag, New York, 2000.

[9] Intel Corp., Mobile Intel Pentium III Processors, Enhanced Intel SpeedStep(R) Technology, *http://support.intel.com/support/processors/mobile/pentiumiii/sb/CS-007522.htm*

[10] Atheros Communications Inc., Power Consumption and Energy Efficiency Comparison of WLAN Products, *http://www.atheros.com/pt/atheros_power_whitepaper.pdf*

[11] Digital Semiconductor, SA-1100 Microprocessor Technical Reference Manual, March 1998.

[12] N. Kamijoh, T. Inoue, C. Olsen, M. Raghunath and C. Narayanaswami, "Energy trade-offs in the IBM Wristwatch Computer", in *Proceedings of the Fifth International Symposium on Wearable Computers*, pp. 133-140, October, 2001.

[13] D. Dunne, "What is a Denial of Service Attack?", *http://www.darwinmag.com/learn/curve/column.html?ArticleID=115*

[14] Military Survelliance, *http://wins.rockwellscientific.com/WST_Survey.html*

[15] M. Fitzgerald, "Intel Inside Healthcare Technology", *http://www.bio-itworld.com/archive/050903/horizons_intel.html*

[16] Body-Sensor Wireless Networks, Monitoring patients' health with BAN *http://www.fokus.gmd.de/medizin/download/BAN_E.pdf*

[17] J. Harrison, "Piecing it Together: Interoperability Enhances Planning and Response" *http://www.geoplace.com/gw/2002/0205/0205ogc.asp*

[18] S. Udani and J. Smith, "The Power Broker: Intelligent Power Management for Mobile Computing", *Technical Report MSCIS -96-12*, Department of Computer and Information Science, University of Pennsylvania (1996).

[19] M. Weiser, B. Welsh, A. Demers and S. Shenker, "Scheduling for reduced CPU energy", in *Proc. 1st Usenix Symp. Operating Systems Design Implemenation*, pp. 13-23, November 1994.

[20] Orinoco Inc., Orinoco PC Card User's Guide, *http://www.workingwireless.net/wireless/Documents/_Lucent/orinoco_ug_pc.pdf*

[21] Machintosh News Network, "IDG: Laptop users adopting wireless networks", *http://www.macnn.com/news/20669*

[22] J. G. Spooner, "Notebook sales hit new high", *http://news.com.com/2100-1044_3-1022905.html?tag=st_rn*

[23] J. G. Spooner, "Handheld market gets holiday gift", *http://news.com.com/2100-1041-5148252.html?tag=cd_top*

[24] T. Martin, E. Jovanov and D. Raskovic, "Issues in Wearable Computing for Medical Monitoring Applications: A Case Study of a Wearable ECG Monitoring Device", *Proceedings of the 2000 International Symposium on Wearable Computers*, pp. 43-50, Oct 2000.

[25] E. Hansen, "Public Schools: Why Johnny can't blog", *http://news.com.com/2009-1023-5103805.html*

[26] F. Tiboni, "Third Infantry buys new rugged laptops", *http://www.fcw.com/fcw/articles/2003/1110/web-laptops-11-14-03.asp*

[27] P. Koopman, *www.ece.cmu.edu/~ece548/hw/lab1/cachesize.c*

[28] CERT Co-ordination Center, *http://www.cert.org/tech_tips/denial_of_service.html#4*

[29] T. Martin, M. Hsiao, D. Ha and J. Krishnaswami, "Denial of Service Attacks on Battery Powered Mobile Devices", in *2nd IEEE International Conference on Pervasive Computing and Communications*, March 2004, to appear.

[30] C. P. Pfleeger, *Security in Computing*, 3rd edition, Prentice-Hall, December 2002.

Vita

Jayan Krishnaswami was born in Ahmedabad, India. He earned his Bachelors of Engineering (B.E) from Nirma Institute of Technology affiliated with Gujarat University in 2001, with Electronics and Communication as his major. He joined the Bradley Department of Electrical Engineering of Virginia Tech in 2001 as a Master's student. In the second year of his Masters program he started working with Dr. Tom Martin on his thesis. Jayan's hobbies include cricket, music and reading. His research interests include wireless communications, digital signal processing and low power mobile computing.