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

Thomas Martin, Michael Hsiao, Dong Ha, Jayan Krishnaswami
Virginia Tech, Dept. of ECE
{tlmartin, mhsiao, ha, jkrishna}@vt.edu

Abstract

Sleep deprivation attacks are a form of denial of service attack whereby an attacker renders a pervasive computing device inoperable by draining the battery more quickly than it would be drained under normal usage. We describe three main methods for an attacker to drain the battery: (1) Service request power attacks, where repeated requests are made to the victim for services, typically over a network--even if the service is not provided the victim must expend energy deciding whether or not to honor the request; (2) benign power attacks, where the victim is made to execute a valid but energy-hungry task repeatedly, and (3) malignant power attacks, where the attacker modifies or creates an executable to make the system consume more energy than it would otherwise. Our initial results demonstrate the increased power consumption due to these attacks, which we believe are the first real examples of these attacks to appear in the literature. We also propose a power-secure architecture to thwart these power attacks by employing multi-level authentication and energy signatures.

1. Introduction

The ongoing proliferation of battery-powered computing devices has created a new type of "denial of service" attack: If an attacker can drain a device's battery, for example, by having it repeatedly execute an energy-hungry program, the device will be rendered inoperable, which Stajano and Anderson have called a *sleep deprivation torture attack, or battery exhaustion* [18]. Unlike other denial of service attacks where the attacker must keep up the attack in order to continue to deny the service, the attacker can quit attacking a battery-powered device once she has fully discharged the battery, and can then move on to attack another device. Just as the advent of computer networks enabled an increase in the number of computer viruses, Trojan horses, and other computer security breaches, the rising availability and increasing dependence on pervasive computing devices will lead to the creation and spread of sleep deprivation torture attacks. The battery in a pervasive computing device is thus a point of vulnerability and must be protected.

Pervasive computing devices, e.g. notebook computers, personal digital assistants, and wearable computers, are intrinsic to making information available anywhere at anytime. There is an increasing reliance on such devices in education, business, emergency response, defense, and healthcare [11], and with increasing reliance comes an increasing need for security [14][23]. While much is known about how to protect pervasive computing devices from attacks on vulnerabilities they share with other types of computing systems, little is known about how to protect them from power-based attacks. Sleep deprivation attacks are not presently widespread, but the potential exists for them to become more common. One of the goals of this paper is to raise the awareness of the pervasive computing community, so that the mechanisms for preventing and mitigating these attacks can be put in place before they become widespread. The attacks that we present here, if not thwarted, could potentially keep pervasive computing from being adopted by a wider group of users.

Sleep deprivation attacks render a device inoperable by draining the battery more quickly than it would be under normal usage. In a typical mobile computer, the battery is expected to give a certain battery life under a set of usage conditions where the user is actively using the device for a small fraction of the time, and the device is idle the rest of the time. When the device is idle, power management software puts the device into low power standby and sleep modes, extending the device's battery life. If an attacker can prevent the device from entering low power modes by keeping it active, the battery life can be drastically shortened. This paper defines three main methods for an attacker to drain the battery: (1) *Service request power attacks*, where repeated requests are made to the victim for services, typically over a network--even if the service is not provided the victim must expend energy deciding whether or not to honor the request; (2) *benign power attacks*, where the victim is made to execute a valid but energy-hungry task repeatedly, and (3) *malignant power attacks*, where the attacker modifies or creates an executable to make the system consume more energy than it would otherwise. Our initial results demonstrate the increased power consumption due to these attacks, which we believe are the first real examples of these attacks on general purpose mobile computers in the literature. Our results show a baseline of what the effect of the attacks may be and the forms that they may take. We expect that the sophistication and impact of these attacks may increase over time, as has happened with conventional worms and viruses, especially if the methods for mitigating the sleep deprivation attacks are added piecemeal to pervasive systems rather than designed in from the beginning.

As will be described in section 3, data presented in [5] shows that an attacker could cut the battery life of currently available mobile computers by a factor of 30 to 280. To put these numbers in perspective, if the expected battery life of a device is a month, an attacker could reduce the life to between 2.5 hours and 24 hours. An attacker could thus leave the victim without access to the device much sooner than expected, reducing the battery life by *one to two orders of magnitude*.

The remainder of this paper is organized as follows. Section 2 describes the related work. Section 3 discusses the potential forms and impact of the sleep deprivation attacks on general-purpose mobile computing systems. Section 4 describes our initial implementations of each form of the attack, our experimental setup, and our experimental results. Section 5 proposes a *power-secure architecture* that we believe will mitigate the impacts of the attacks. While this architecture is not yet fully implemented, we outline its major features. Section 6 provides our conclusions and possible avenues for future investigation.

2. Related Work

The work most closely related to this paper includes sleep deprivation attacks on sensor networks, power analysis of encryption devices, authentication in distributed environments, low power software design, and peak power estimation.

To the best of our knowledge, the first mention in the research literature of rendering a battery-powered device inoperable by draining its battery has been by Stajano and Anderson [18]. There has been no systematic study of the attack, methods for preventing it, or implementations of it. The main interest in it has come from the wireless sensor network community [16]. A major difference between wireless sensor networks and general purpose mobile computing is that the power consumption of wireless sensors is dominated by the RF subsystem, so the focus there has been on limiting communication in order to thwart the attack. In a general purpose mobile computing device, limiting communication will not prevent all forms of the attack. General purpose mobile computing devices also offer a much richer set of services than wireless sensor networks, and thus there are more forms for the attack to take on them. Thus it is necessary to study the attacks on mobile computers to find ways to mitigate these other forms.

Another seemingly major difference is that wireless sensor networks are typically unable to have their batteries recharged or replaced, and thus the sleep deprivation attack would appear to be more critical for them than for mobile computers. However, in practice, the attack on a mobile computer would typically be just as critical in terms of denying its use. A mobile computer subjected to a sleep deprivation attack would likely not be able to have its battery replaced or recharged without the user stopping his or her current activity, finding a place to recharge the battery (assuming the charger has been brought along with the mobile computer), and then waiting until the battery is recharged. For users in the field, e.g. emergency response workers, military personnel, etc., recharging or replacing the battery may not be an option, and thus the sleep-deprivation attack on mobile computers is just as critical as it is on wireless sensor networks. For consumer applications, if an attack were not detected, the user would probably suspect that the battery of the device had become defective and was no longer able to retain any charge.

One security issue involving power that has been widely studied is that of *power analysis of encryption devices.* Differential power analysis (DPA) attacks the key of a cryptosystem by measuring power consumption during encryption/decryption and finding the correlation between the power consumption and the value of bits of the key [6][17]. DPA has been demonstrated to reveal substantial portions of a key, for example, 48 bits of a 56-bit DES key. However, unlike sleep deprivation attacks, DPA is an attack on the confidentiality of a device's key, not the availability of the encryption device itself.

The authentication process we envision for the proposed power-secure architecture shares many aspects with techniques for authentication in a distributed environment, such as Kerberos [7] or X.509 [4]. Kerberos depends upon the constant availability of a centralized authentication server, which is not applicable or desirable for the wireless networked environment. In contrast, X.509 depends upon cryptographically signed certificates from a certificate authority; the authority does not need to always be available. For battery-powered devices, authentication based upon X.509 may be applicable if the encryption algorithms and protocols are evaluated for their energy usage

Research in low power design is also applicable to the problem, particularly in estimating battery life of mobile systems [12], measuring power consumption of software [1][19] and in creating low power software

[13]. When services that consume too much energy are identified, it will be desirable to reduce their power consumption via low power compilation and source code transformations. One major difference between attacks on batteries and low power design has is that the goal of low power design is typically assumed to be to lower the energy per operation of the device, which is a measure of both the power consumption and the execution time of an operation. For attacks on the battery, the attacker's goal will be to maximize power consumption, without regard to the number of operations that are performed in a given amount of time.

In regard to peak (or maximum) power, if the peak power is significantly higher than average power of the device, the battery can be depleted at a faster rate if the attacker can sustain the peak power consumption. Estimation of peak and maximum power has attracted attention in recent years [10][22], in which an attempt to place a bound on peak power dissipation is made. Taking the peak power further by one step, sustainable peak power generation that attempts to find a cycle with peak power is generated can be used to generate a power attack to test our system [2]. New methods for computing sustainable peak power are developed in which automatic generation of a functional vector loop for near-worst case power consumption is attained. The peak power measures obtained more than 70% tighter peak powers have been achieved [2].

3. Potential Forms and Impact of Sleep Deprivation Attacks

We define three main forms of sleep deprivation attacks on general purpose mobile computers: Service request attacks, benign power attacks, and malignant power attacks. The goal of each type of attack is to maximize the power consumption of the target, thereby decreasing its battery life. The attacks achieve this by keeping the target device busy, and preventing it from going into low power sleep modes. However, the mechanism for each attack is different:

(1) *Service request power attacks* repeatedly make otherwise valid network service requests, such as telnet, ssh and web server requests, for the purpose of using up the device under attack's (DUA) battery capacity. This type of attack keeps the DUA busy authenticating/servicing the requests.

(2) *Benign power attacks*, where the DUA is made to execute a valid but energy-hungry task indefinitely, such as displaying a hidden animated gif or executing a hidden Java script; though invisible to the user, the task secretly drains the energy source. The essential feature of the benign power attack is that the attacker provides data to a valid program that causes the program to execute in such a way that it consumes a pathological amount of power.

(3) *Malignant power attacks*, where the attack maliciously penetrates the system and alters operating system kernel or application binary code such that more energy is needed to execute them; the altered binaries may or may not be functionally correct. These attacks will thus be either viruses or Trojan horses.

In some cases, these attacks can be prevented or detected using existing security techniques. For example, malignant power attacks can be detected using virus-scanning software. But in other cases, for example, the benign power attacks, detecting the attacks using existing techniques will be difficult. There is a chance that the security techniques could themselves be used to mount a sleep deprivation attack: An attacker could send a virus that he knows will be caught by the target system's virus-scanning software, but the energy consumed by the virus-scanning software may exhaust the battery if the attacker causes it to run repeatedly.

A successful attack will maximize power consumption while presenting to the user the appearance that the system is behaving normally, with the possible exception of the battery status indicator. Side effects that one would expect to see of these attacks if they are not implemented subtly include the CPU fan turning on while the user is performing some action that does not normally cause the fan to come on, the system becoming less interactive than usual, and the hard drive spinning up immediately after a spin down. A successful attack will likely cause the user to believe that the battery has become defective and will no longer keep a charge. We outline an architecture for detection in section 5 that we believe will detect all three forms of power attacks, even if the attacker is careful about not causing otherwise suspicious side effects.

Another property of a successful attack is that it will utilize the subsystems that have the largest difference between idle or sleep state power consumption and active state power consumption. In the systems we present in Section 4, this was the CPU. Thus our implementations of the attack tended to exercise it heavily. For other platforms, different subsystems may be more attractive targets.

To illustrate the potential of these attacks, assume that the device uses power P_{active} while active and power P_{sleep} while sleeping, that $PFR = P_{active}/P_{sleep}$, and that the device has a usage duty factor of D (fraction of time that the device is active) [5]. Then the battery life, normalized to being asleep 100% of the time ($D=0$), is equal to $1/(1-D + PFR \times D)$. Since PFR is much greater than 1, in order to minimize battery life, one should increase *D*, increase *PFR*, or both. Typically *D* is very small; the value reported in [5] was 0.0035, which is equivalent to using a device 10 times a day for about 30 seconds at a time. Reported values of *PFR* for a range of PDAs and experimental mobile computers range from 30 for the experimental platforms (Compaq Itsy and IBM Linux Wristwatch) to 280 for the commercially available PDAs (Psion and Palm Pilot) [5].

The aim of the attacks is to keep the device as active as possible, to make *D*=1. Assuming that in normal usage D is very small, then the battery life when under attack will be reduced by a factor of approximately *PFR*. Given the range of values for *PFR* from above, an attacker could reduce the battery life of currently available mobile computers by a factor of 30 to 280. Our experience with various notebook computers in the laboratory shows that the *PFR* for them is much smaller, on the order of 2 to 4, because they tend to be in the idle state rather than the sleep state when not active. But even with these much smaller values of PFR, the battery life would be considerably shortened.

In some circumstances the impact of the attack may be worse. The above analysis ignores non-ideal capacity properties of batteries that may make the attack even more effective, because it implicitly assumes that battery life is inversely proportional to the average power. In practice, the battery energy capacity depends upon the power consumption of the load [12][15]. As the power of the load increases, the total energy that is delivered by the battery decreases. Over a range of loads that can be reasonably expected of a mobile computer, the battery capacity may vary by nearly 30% [12]. What this means in practice is that, if an attacker can increase the average power by a factor of X, the battery life will be shortened by more than a factor of approximately $X^{1+\alpha}$, where α is in the range of 0.2 to 0.7 [9].

4. Power Attack Examples

This section gives examples of each of the three types of power attack 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, we first describe our experimental setup.

4.1. Experimental Setup

Data for the power consumption of the mobile devices was collected using an Agilent 3458A digital multimeter set to a sampling rate of 10,000 samples/second. The multimeter was controlled by a computer that also stored the readings taken, using a triggering arrangement similar to that described by Flinn [1]. This arrangement allowed us to synchronize measurements over several trials and average them together.

For the measurements, we used a notebook computer and two different PDAs for 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 [21]. The IBM Thinkpad has a 866MHz Pentium III CPU, 128 MB of memory and dual boots the Windows 2000 and Linux operating systems. The iPAQ is a 3675 model, with 64 MB of memory, running Windows CE version 3.0. The Itsy is 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 we used an Orinoco Silver 802.11b wireless network card to provide a network connection to them for the service request attack results.

For the IBM Thinkpad and the iPAQ, the screen brightness was a large factor in the power consumption; to provide consistent behavior from one trial to another, screen brightness was set to its minimum value during all measurements. We also ensured that power management settings were the same for each trial.

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. A program was written that ‘sleeps’ for a fixed amount of time before giving out a pulse on a port that, in turn, was connected to the multimeter’s external trigger input. This helped to begin the applications on the systems within a couple of milliseconds of each other over many runs. The plots in this section show the power as a running average of 100 samples (10 ms) in order to filter out high frequency noise that makes it difficult to see the general trends.

4.2. Description of the attack implementations

We now describe some specific attacks that we have implemented, attacks that can be used on a variety of platforms. As we said earlier, these examples are not meant to be exhaustive, but are meant to demonstrate the existence of the attacks and to give some indication of their potential impact. Furthermore, we are not claiming that these are the most potent forms of the attack. While the examples we present here are straightforward and easy to mount, we suspect that there are more subtle vulnerabilities that can be exploited. For example, all of our attacks change the state of the system from what would generally be considered an idle state, where the device appears to be on but is not actively performing any activity, to an active state. More powerful forms of the attack would have the device appear to be in a standby state (which typically consumes much less power than being in the idle state) while in reality being active, e.g. having the screen off as if in standby but having a thread that is being executed on the CPU. The important point is that these are the first demonstrations of sleep deprivation attacks on general purpose mobile computers to appear in the literature.

For the service request attack, we used repeated requests to an SSH server. The requests were made with a correct username but a wrong password. For the benign attack, we created an animated GIF that consisted of the same image shown repeatedly; to the eye it appeared to be unanimated. To provide a comparison point, we also measured the power consumption of an unanimated version of the same image.

Finally, 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 . As section 4.3 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 tune it to the particular target system.

We tried all three attacks on both the IBM Thinkpad and the iPAQ, but because of the network and screen limitations of the Itsy we did not try the network service request or the benign power attack on it. For the Itsy, we tried only the malignant attack; due to some interesting effects of the CPU speed on power consumption, and because the Itsy was designed to allow power to be measured to both the CPU core as well as the entire system, we include the Itsy results for this attack because it allowed us to test our hypothesis about the CPU speed and its impact on the relative power cost of cache hits and cache misses.

4.3. Experimental Results

Service request attack: Figure 1 shows the results of the service request attack on the IBM Thinkpad running an SSH server. The Y-axis denotes the power dissipated as a function of time (X-axis). As shown in Figure 1, the idle power (first five seconds) of the Thinkpad is approximately 10W. When the repeated denied service requests begin, the results show that the average power consumption increases by approximately 15%. Figure 2 shows the results of the same attack on the iPAQ running the SSH server. (Please note the difference in scale of the y-axis between Figures 1 and 2.) Here the results are more dramatic, with about a 30% increase in average power consumption while the failed logons are taking place.

Our initial form of the attack on the iPAQ used an FTP server but had relatively little power increase, about 5%. This was much lower than we initially expected, because on the iPAQ the 802.11b card is a large fraction of the overall system power. Given the emphasis on this attack in the sensor network area, we assumed that the transmit and receive power of the 802.11b card would dominate, but in fact the power consumption of the card while maintaining the wireless connection (approximately 1.2W) is only slightly less than while actively transmitting and receiving. Thus the increase in power consumption is mainly due to the computation required by the requested service rather than the wireless communication, which is a major difference between this form of the attack on a wireless sensor network. If

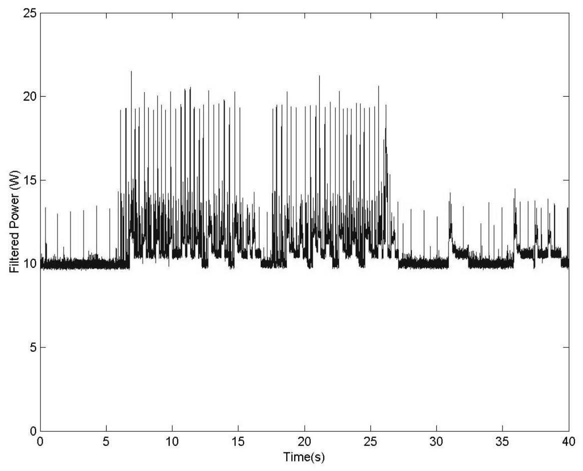

Figure 1. Requests made to SSH server on the Thinkpad with wrong password.

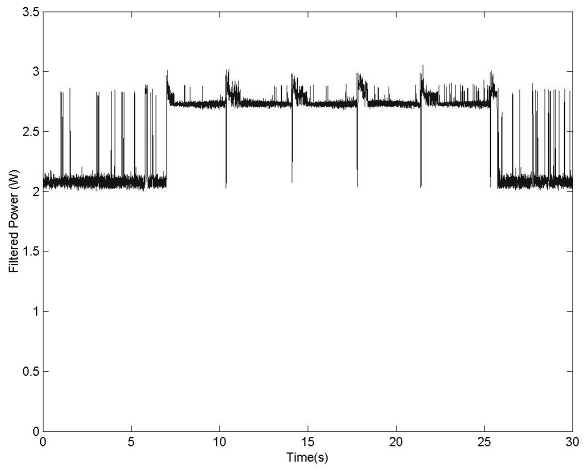

Figure 2. Requests made to SSH server on the iPAQ with wrong password.

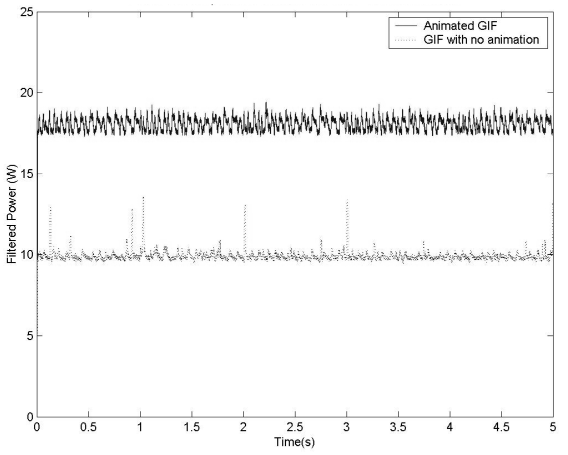

Figure 3. Comparison of power consumption of animated GIF and non-animated GIF on the Thinkpad

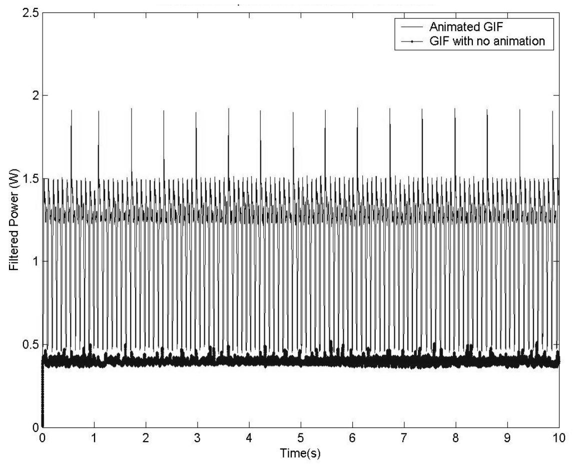

Figure 4. Comparison of power consumption of animated GIF and non-animated GIF on iPAQ

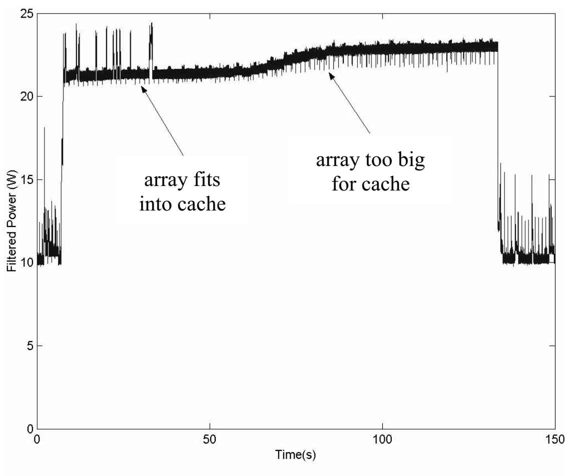

Figure 5. Power consumption during malignant power attack on the Thinkpad

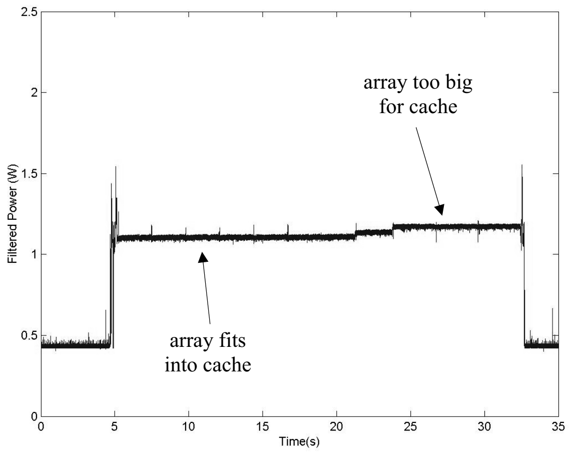

Figure 6. Power consumption during malignant power attack on the iPAQ

the power management of the 802.11b card were such that there was a larger difference between maintaining a connection and actively receiving/transmitting, then this attack would cause a larger increase in power consumption. These results also show that, for general purpose systems at least, the computation required by a network service request should be limited to mitigate the potential for this form of the attack. In Section 5 we will address this by using multi-layer authentication for service requests.

Benign power attack: The results of the benign power attack on the Thinkpad are shown in Figure 3. The animated GIF consumes about 80% more power than a non-animated GIF (18.4W vs. 10.2W). This is significantly higher than a non-animated image, even though to the human eye the animated version appears to be the same. Figure 4 shows the benign power attack results for the iPAQ; the animated image give consumes almost three times more power than the unanimated image, 1.15W for the animated image vs. 0.43W for the unanimated image. (The average power for the iPAQ is lower than in the service request attack results because the 802.11b card was not present.)

An attacker could thus embed an animated GIF in a web page in such a way that the user would think that it was unanimated while causing the user's device to consume considerably more power than if the image were actually unanimated. The attacker does not need to modify the target system's software or hardware configuration in any way, nor does the attacker have to actively make a network connection to the target. Instead the attacker provides pathological data to the existing software on the system such that the system consumes more power than it would under normal usage. This attack is probably the most difficult of the three attacks to defend against.

Malignant power attack: Figures 5 and 6 show the malignant power attack on the Thinkpad and the iPAQ, respectively. For the Thinkpad, the attack consumes 21.4W while hitting in the cache and 22.9W while missing in the cache, more than double the idle power of 10.2W. For the iPAQ, the attack consumes 1.10W during cache hits and 1.17W during cache misses, about three times the idle power of 0.42W. One feature of our implementation of the malignant power attack is that on both the Thinkpad and the iPAQ, cache misses consume more power.

For the Itsy, however, the situation is more interesting, as shown in Figure 7 and Figure 8. The Itsy research prototype has the option to allow the user to change the CPU clock frequency. The malignant power attack was run on the Itsy for two different processor speeds (206 MHz and 59 MHz). At 206 MHz, a cache hit consumes about 10% more power than a cache miss; at 59 MHz a cache miss consumes about 10% more than a cache hit.

The plots in figures 5-8 show that if an attacker tries to drain the battery by exploiting the cache behavior, he must know about the particular hardware of the target. The hardware configuration of the device under attack determines whether a hit or a miss in the cache consumes more power. One option is to maintain an exhaustive database of all the devices and their clock frequencies, which seems impractical. A more practical option is for the attacker to get the power information from the battery state.

We have written an initial version of a self-tuning malignant power attack on the Thinkpad. It uses the battery information provided by the Thinkpad's Advanced Configuration and Power Interface (ACPI) subsystem [3] to monitor power consumption while the attack is underway. Once the array does not fit into the cache, the program chooses the array size that had the most power consumption. The power consumption provided by ACPI battery interface is coarse; the power is sampled about once a second. However, by allowing each size of the array to be accessed for about 30 seconds at a time, we are able to get adequate power information from the battery to distinguish whether hitting in the cache or missing in the cache consumes more power.

As was stated in Section 3, if the system becomes noticeably less interactive, the user may become suspicious. To avoid reducing the system's interactivity, our self-tuning malignant attack runs at the lowest priority, such that the only the idle process has lower priority than it. Thus it will run whenever the idle process would normally run, keeping the CPU active, without interfering with the performance of the user's other running programs.

5. Towards a Power-Secure Architecture

In order to guard against attacks that attempt to quickly drain the device's energy source, we are currently implementing a hardware/software architecture for thwarting sleep deprivation attacks on general purpose mobile computing systems, which we call a *power-secure architecture.* The goal of the architecture is to provide some guaranteed fraction of the system's expected battery life. This section gives an overview of the architecture.

The overall power-secure architecture employs two fundamental security features within the system: multi-layer authentication and energy signature monitoring. The multi-layer authentication is designed to prevent energy loss from service request attacks by making sure that all untrusted services rendered consume less than a certain amount of energy. Additional resources are committed only to those requesters who have obtained further levels of trust. The energy signature monitor, which requires a self-contained unit to measure the system's dynamic power and a database of reference signatures for each trusted application, is designed to catch intrusions that have entered the system to execute an energy-hungry application or service, either as a malignant or benign

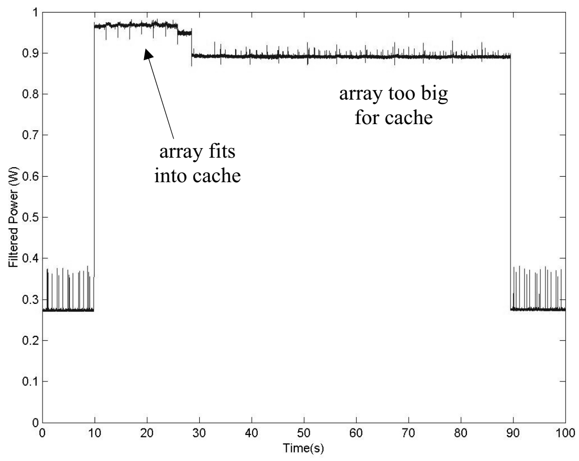

Figure 7. Power consumption during malignant power attack on the Itsy at 206 MHz

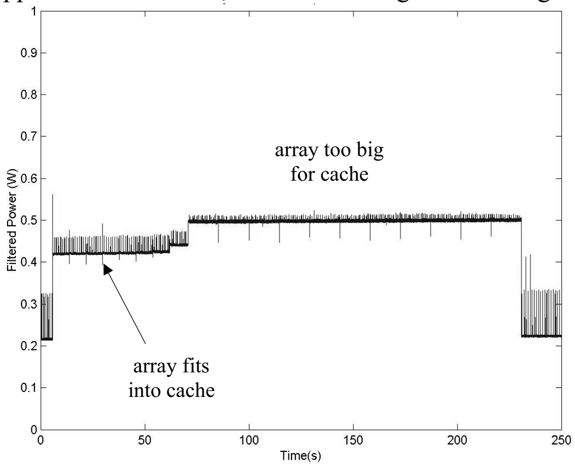

Figure 8. Power consumption during malignant power attack on the Itsy at 59 MHz

power attack.

5.1. Multi-layer authentication

In the event of a service request attack, repeated requests are made by the attacker to keep the device under attack (DUA) extremely busy. If the service requires authentication, such as telnet/ssh, traditional authentication approach will always perform a database lookup to match the password with username, thus drawing unnecessary current in the process. Likewise, services requiring no authentication, such as web page requests, can be used to force the DUA into thrashing mode, feverishly responding and servicing the requests, possibly denying regular and normal requests in the process.

Figure 9 illustrates our authentication technique. The energy profile of a service is generally non-linear over the lifetime of the service as depicted in part (a) of the figure, thus repeated executions of one small portion of the service may consume less energy than if the entire service was repeated continuously. For example, if we allow the first T seconds to be executed before any authentication, the power profile under an attempted attack will be similar to that shown in part (b) of the figure, where the average power consumption is kept nearly constant. On the other hand, if no measure is taken to prevent such attacks, the profile would look similar to part (c) of the figure, where repeated calls to the service drains a substantially more energy. To find the time T at which a service request should be authenticated, for each service we calculate a *crippling energy level*.

5.2. Crippling energy level

The crippling energy level is the worst-case repeated-access to a service that can cripple the device in a given amount of time. This pre-determined crippling energy, e_c, for each service is computed in the following manner. Let us assume the total energy available for the device is E, and we would like to have a minimum lifetime of the device to be at least L. In order to guarantee that the device lifetime is greater than L, we must make sure that repeated requests to a particular service consume no more than E energy in L. Because the energy consumption of the entire service may be non-linear over the lifetime of the service, servicing repeatedly a small portion of the service may consume less energy than repeatedly servicing the entire service. Thus, if the energy consumption for the first T seconds is e_c, then we must ensure that $E/(e_c / T) >= L$, or $e_c <= E \times T/L$.

One difficulty with this authentication approach is that we must guarantee that repeated authentication (the first initial portion of the executable) would not reduce the lifetime of the device to be less than L. Otherwise an attacker can exhaust the battery simply by repeatedly forcing the authentication to occur. Therefore, not only must this initial authentication be a lightweight process, it must also satisfy the crippling energy criterion. It may not be possible to have a single, lightweight authentication that is sufficiently

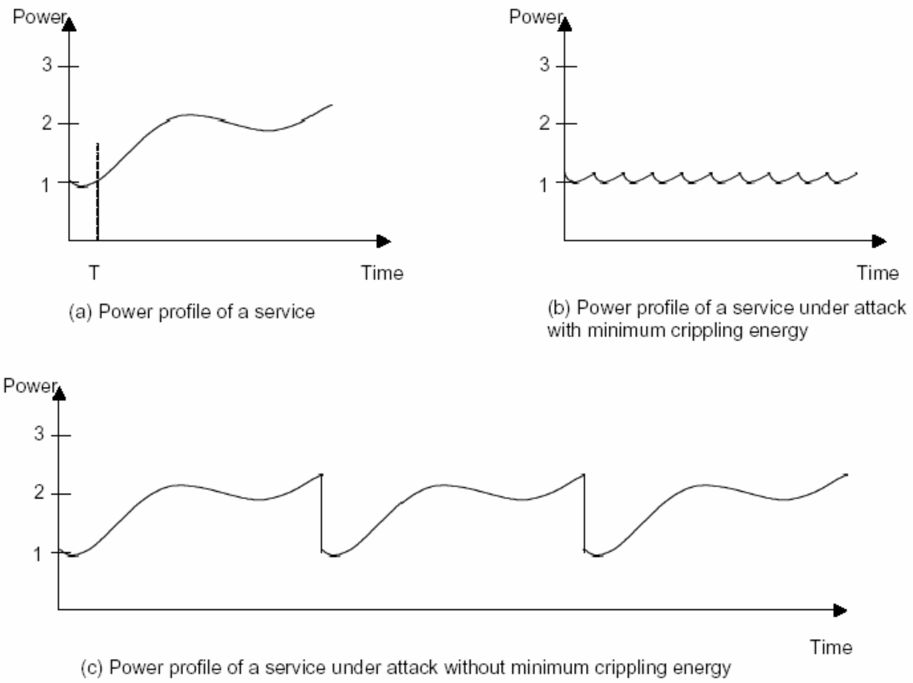

Figure 9. Power profiles of a service showing the concept of the crippling energy level

difficult to defeat. Thus we may have to use multiple layers of authentication, with services that have higher energy requirements going through several stages of authentication as they run, with each layer possibly requiring more energy.

If a service can complete before it consumes the crippling energy, then it does not need to be authenticated: Even if it were run continuously, the device would still have a battery life greater than L. Once a service continues beyond the initial authentication, the energy signature of the service is captured dynamically and is validated against the reference signature. If the validation fails, the system is considered under a power attack.

5.3. Energy Signature Monitor

If an attack has somehow successfully intruded the system (the attacker has been authenticated), the power-secure architecture remains vigilant by continuing to monitor energy consumption of the system to make sure it is within expected bounds of the application. Benign and malignant power attacks would alter the dynamic energy signature of the particular service. We will detect such intrusions via dynamic validation of the dynamic energy signature against known energy signatures for the device and application, and the lifetime of the system under attack will be protected. The energy signature monitor relies of a self-contained unit for dynamically measuring the systems power consumption, the energy monitoring unit (EMU), similar to what is available via ACPI but with greater fidelity. The EMU should permit the system to measure the energy consumed of each process. The energy measurements made by the EMU will be compared to a set of reference energy signatures to detect whether the system is under attack. For most programs, it will be difficult (if not impossible) to generate a signature for every possible valid execution of the program. Even if it were possible to generate signatures for every possible valid execution, storage constraints of mobile computing devices will limit the number that can be carried on the device. Thus an open issue for this signature method is finding a concise representation of the signatures that provides an adequate level of accuracy.

One concern that we have for the EMU is that the power measurements it makes available would allow an attacker to more easily tune a sleep deprivation attack. Thus accessing the EMU must be a privileged operation. Another concern is that, similar to multi-layer authentication, checking the signatures must be a lightweight process and cannot consume too much energy. Consequently, we suspect there will be a trade-off between the accuracy of the method used to check the signatures (i.e., the rate of false positives and negatives) and the amount of energy and storage it requires. One way to reduce the energy cost of checking signatures is to use some method of intrusion detection that looks for sleep deprivation attacks and only check signatures when an attack is suspected. For example, signatures could be checked when the power consumption becomes large enough that the battery life will be less than the guaranteed minimum. Then signature checking will only occur when the predicted battery life is suspiciously short. We are investigating methods of intrusion detection that are appropriate for sleep deprivation attacks, while meeting the power, performance, and storage constraints of battery-powered systems.

6. Conclusions

This paper has described sleep deprivation attacks on general-purpose battery-powered computing devices. These power-related security attacks render a device inoperable by draining the battery more quickly than it would be under normal usage. If an attacker can prevent the device from entering low power modes by keeping it active, the battery life can be drastically shortened. We have defined three main methods for an attacker to drain the battery: (1) Service request attacks, where repeated requests are made to the victim for services, typically over a network--even if the service is not provided the victim must expend energy deciding whether or not to honor the request; (2) benign power attacks, where a the victim is made to execute a valid but energy-hungry task repeatedly, and (3) malignant power attacks, where the attacker modifies an executable to make it consume more energy than it would otherwise. Our initial results show that denial-of-service attacks prevent the Thinkpad and PDA from entering low-power sleep/idle modes, while at the same time consuming significant additional power. As far as we know, these are the first implementations of sleep deprivation attacks to be reported.

One very interesting result is that the potential impact of service request attacks on general purpose systems is determined by the computational requirements of the requested service, not by the power of the wireless communication subsystem, unlike wireless sensor networks. When the additional power consumed by actively transmitting/receiving on the wireless network interface is small, the service request attacks will greatly increase power consumption only when the requested service has a large computational requirement.

Our results also demonstrated that the cache performance of an attack alters the power profile. Thus a smart attacker will create attacks that can tune themselves to increase their power consumption on a target system. System designers should be aware that information provided by the power management subsystem with the intent of increasing battery life could be used by an attacker to tune a sleep deprivation attack. For example, our self-tuning malignant power attack uses battery information provided by ACPI to tune itself. ACPI also provides information about other subsystems, their power management states, and the power consumed in each state, which could be used by an attacker to choose subsystems that will give the largest increase in power if they were kept in an active state. If sleep deprivation attacks proliferate then access to the power management subsystem must be privileged.

The paper also presented our proposed power-secure architecture. The architecture will defend against these attacks by guaranteeing a minimum battery life even in the face of the attack. Multi-layer authentication and energy signature monitor form the basic pillars for the architecture. The multi-layer authentication is designed to prevent energy loss from service request attacks by making sure that all services rendered consume less than a certain amount of energy. Additional resources are committed only to those requesters who have obtained further levels of trust. The energy signature monitor is designed to catch those intrusions that have entered the system to execute an energy-hungry application or service. An initial implementation of the power-secure architecture is underway. Our future work includes fully implementing the architecture and testing it on a variety of systems.

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

References:

[1] J. Flinn, and M. Satyanarayanan, “PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications,” 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, Feb. 25-26, 1999, pp. 2-10.

[2] M. Hsiao, "Peak power estimation using genetic spot optimization for large VLSI circuits," Proc. of the IEEE Design, Automation, and Test in Europe Conference, Mar., 1999, pp. 175-179.

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

[4] ITU-T Rec. X.509 (Revised), “The Directory-Authentication framework, International Telecommunication Union, Geneva, Switzerland, 1993.

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

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

[7] J. Kohl, B. Neuman, and T. Ts’o, “The Evolution of the Kerberos Authentication Service.” In Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994.

[8] H. Kriplani, F. Najm, P. Yang, and I. Hajj, "Resolving signal correlations for estimating maximum currents in CMOS combinational circuits," Proc. Design Automation Conf., 1993, pp. 384-388.

[9] D. Linden, Handbook of Batteries, 2nd ed. New York: McGraw-Hill, Inc. 1995.

[10] S. Manich and J. Figueras, "Maximizing the weighted switching activity in combinational CMOS circuits under the variable delay model," Proc. European Design and Test Conf., 1997, pp. 597-602.

[11] 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, Atlanta, GA, Oct. 2000, pp. 43-50.

[12] T. Martin, and D. Siewiorek, "Non-Ideal Battery Behavior and Its Impact on Power Performance Trade-offs in Wearable Computing," Proceedings of the 1999 International Symposium on Wearable Computers, San Francisco, CA, October 18-19, 1999; pp. 101-106.

[13] P. Ong, and R. Yan, “Power-Conscious Software Design--a framework for modeling software on hardware,” Proceedings of the 1994 Symposium on Low Power Electronics, October 1994, pp. 36-37.

[14] M. Satyanarayanan, "Pervasive computing: vision and challenges," IEEE Personal Communications , Volume: 8 Issue: 4 , Aug. 2001, Page(s): 10 -17

[15] T. Simunic, L. Benini, L. G. de Micheli, “Energy-efficient design of battery-powered embedded systems,” Proceedings of the International Symposium on Low Power Electronics and Design, Aug. 1999, pp. 212 -217.

[16] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, M.B. Srivastava, M. Potkonjak, "On Communication Security in Wireless Ad-Hoc Sensor Networks", IEEE WETICE 2002.

[17] N. Smart, "Physical side-channel attacks on cryptographic systems," Software Focus, 2000, vol. 1, no. 2, pp. 6-13.

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

[19] P. Stanley-Marbell, and M. Hsiao, "Fast, flexible, cycle-accurate energy estimation," Proceedings of the ACM/IEEE International Symposium on Low-Power Electronics and Design, August 2001, pp. 141-146.

[20] V. Tiwari, S. Malik, and A. Wolfe, “Compilation Techniques for Low Energy: An Overview,” Proceedings of the 1994 Symposium on Low Power Electronics, October 1994, pp. 38-39.

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

[22] C. Wang, K. Roy, and T. Chou, "Maximum power estimation for sequential circuits using a test generation based technique," Proc. Custom Integrated Circuits Conf., 1996.

[23] M. Weiser, "Some computer science issues in ubiquitous computing," Communications of the ACM, Volume 36, Issue 7 (July 1993).