Possible vectors to attack energy

Adam Hitchcock

April 7, 2008

Abstract

This paper examines the relatively unexplored area of energy security. I will discuss many ways to attack the energy supplies of mobile and wireless devices as well as servers and large scale data centers. Nearly all *Energy Attacks* have a long lasting effect even after the attack ends. This effect can be physical such as a drained battery or fried circuitry, or the effect can be fiscal due to the greatly increased cost of operation during the energy attack. I will also present my experimentation on attacking laptops where I was able to reduce a normally six hour battery down to one and a half hours.

1 Introduction

As computing moves forward technology becomes smaller and more agile. In recent years this has prompted a shift to mobile computing in the form of laptops, cell phones, hand held computers, wireless sensors, and more. This mobile demographic has additional security concerns besides those which typically plague Internet connected devices. These new vulnerabilities are related to the energy performance requirements of this new generation of computers. While there has been a large amount of research analyzing how to increase energy efficiency of various systems there has been little research on the security issues related to energy.

In this paper, I will discuss the notion of an *Energy Attack*, that is an attack which aims to increase power consumption on a device. The intent behind an energy attack can be to drain a battery or to run up the fiscal cost of operation. Such an attack has obvious consequences for emergency situations, such as disaster relief, but the attacks cary less obvious economic consequences for non-critical industries. Also interesting about these attacks is the high likelihood for them to go unnoticed until it is too late.

Ultimately I will report on my experimentation of one such attack. The attack is implemented using DAAP — Digital Audio Access Protocol — to rapidly stream media off a victim's hard drive. Especially troubling is that no particular vulnerability is exploited in the attack, it is simply using an open service as it is intended for an ends other than is intended.

The rest of this paper is organized as follows: Energy attacks are defined in section 2, in section 4 I discuss the goals and fundamentals of an energy attack. Section 5.1 and section 5.2 discuss specific vulnerabilities in wireless and wired systems respectively. Delivery and propagation is discussed in section 5.3. In section 6 I talk about the experimentation performed and in section 6.2 I present my results. I conclude in section 9.

2 Energy Attacks

In the previous section we saw how there exist current exploits utilizing flaws in the intersection of the 802.11 protocol and proprietary energy saving mechanisms. They use flaws in the energy hardware to launch a Network Denial of Service attack but do not target the energy supply of the machine. When the attack ends full service is restored these kinds of attacks shall be known as *Energy Related Attacks*. Energy Related Attacks are interesting as they are a recent phenomenon. They are the result of the recent trend towards mobile devices and industry responding by providing products with longer battery life. Unwittingly security holes have been introduced.

Energy Related Attack: An attack in which energy related infrastructure is utilized but the attack does not directly target energy consumption, energy related infrastructure, or the energy supply itself. Like most attacks when an Energy Related Attack is ended normal service is resumed.

These vulnerabilities (those in the energy related hardware as well as others) can also be used to attack the energy source its self. *Energy attacks* differ in that there is a lasting effect after the attack ends. This effect can be physical such as a drained battery or fried circuitry, or the effect can be fiscal due to the greatly increased cost of operation during the energy attack.

Energy Attack: An attack which directly targets energy consumption, energy related infrastructure, or the energy supply itself. When an Energy Attack ends there will be a lasting effect on the victims.

Energy attacks have some interesting and unique properties when compared to other Internet attacks. They are generally hard to detect; unless a victim is actively monitoring their battery level, which is uncommon, they will not realize what is going on. This is because the "normal performance" is not overly compromised. Most attacks, if analyzed, should show their network traffic to be nothing remarkable. This is because Energy Attacks aim at a device, not a network.

Energy attacks come in a variety of flavors. They can target the energy source of a mobile device with the intent of depleting it or they can target a server farm with the intent of running up operating costs. In the most extreme case an energy attack could burn out hardware due to heat or a current overload. To discuss theses different kinds of attacks I would like to introduce two additional terms:

Denial of Energy Attack: An energy attack with the express goal of depleting a local energy supply.

Cost of Energy Attack: An energy attack with the express goal of increasing the fiscal cost of operation of a system.

While there is definite overlap between these terms I believe a subtly will emerge which will make them distinct.

3 Motivation

As of now I see a three fold motivation for studying energy attacks:

1. Global energy consumption: We need to study energy consumption and consider the energy efficiency of code as we are at a point where these decisions are very important for our world's future.

2. Fiscal reasons: If the ability exists for a hacker to maliciously increase the cost of operation of a company or governmental server there is a clear security issue. This is especially true as energy consumption is amongst the fastest growing concerns amongst companies and governments. As this happens the potential for energy to pose a security threat grows.

   Also several businesses now require a mobile workforce equipped with laptops and other electronics. For these businesses energy attacks wold cost them time and money.

3. Mobile Computing: Energy has surpassed processing power and storage space as the greatest constraint for mobile computing. In other words, we no longer ask the question if our laptops are powerful enough to do our work, but if we can do our work before our laptop battery dies.

4. Safety reasons: One of the growing security concerns in today's world is urban disasters be they caused by nature's wrath or malicious intent. One of the chief failures of 9/11 was the response teams had non-functioning Motorolla communication devices. In this case it was a hardware failure, but in the future it could be an energy attack.

5. Military reasons: Let us change gears now to consider those who do not place the same weight behind fiscal and ecological concerns, such as the military. The primary concerns of the military are the security related. This includes the security of the nation, and of the soldiers in the field. Our military is the most technologically advanced in the world, as a result each soldier carries a several pound battery to power an array of devices. If an opposing force were able to render a soldiers technology useless by running out their battery both the soldier's life and mission could be put at risk.

For a combination of the first two reasons Google has announced a several million dollar investment in renewable energy sources for their data centers. While Google is the first company to propose such a plan others will likely follow suit over the next few years.

Lastly mobile computing is in its infancy, every day dozens of new mobile devices are made. We can not yet imagine the implications of energy for all those devices yet to be made.[cut this?]

4 Goals of An Effective Attack

When considering an energy attack there are certain goals to keep in mind. For this section I will consider general attack goals without considering the specifics of implementation, that is some of these goals will be mutually exclusive for varying implementations.

4.1 Exploit the Energy Hierarchy

In modern programming it is common to keep performance in, for instance in C you wouldn't want to access a matrix in a column major fashion as it would thrash the cash (and for large problems, memory). So accessing a matrix in the proper order does not only increase performance but it also reduces energy usage. One reason for this is that it takes less energy to access registers, than it does to access cache, than it does to access memory, than it does to access disk .

Processors can require variable amounts of energy depending on load, and modern processors are capable of scaling their energy usage depending on system needs. When code runs on a processor the amount of energy require to run that depends on what kinds of instructions are used. A common programming practice is to count by powers of two in a loop. Using the pow() function is far more energy intensive than using a bit shift. So if a naïve programmer makes this mistake that code is now vulnerable to an energy attack.

```
for(i=0; i<10; i++){
A[pow(2,i)]++;
}

for(i=0; i<1024; i<<=1){
++A[i];
}
```

Figure 1: A naïve implementation and an energy optimized implementation on the top and bottom respectively.

If a system has a separate (i.e. not on-board) graphics card, it can use a *very* large amount of energy. While difficult to exploit now, Nvidia has released a kit to allow running certain code on graphics cards, making them a target in the very near future. Also important in energy attacks are wireless chipsets. Currently there are a myriad of wireless communication options including 802.11 a/b/g/n and Bluetooth. Soon they will be joined by 802.11i, USB3, and WiMax.

When considering how much energy various operations take it is important to consider the architecture of the system you are working on. Many hand held devices do not have a south bridge processor, or a dedicated memory controller and I/O controller. This means that accessing its ram or persistent storage must be done through the CPU, this means a higher energy cost on all levels.

4.2 Denial of Service without Interruption of service

Unlike traditional DoS attacks the goal of an energy attack is to have a lasting effect over a longer period of time. Due to this longer time requirement it is important to remain covert for maximal effect. For this reason when considering the effectiveness of an energy attack detectability will play an important role.

5 Attacks

5.1 Wireless Attacks

First I will present attacks with the goal of draining a wireless device's energy supply. These attacks are similar to traditional denial of service attacks in that a service (usually the network) is denied, during the attack other services may remain operational. Energy attacks are unlike traditional attacks as no service is necessarily denied during the attack (more on this later), but when the attack successfully completes no services will be accessible on the device. An partially completed attack will have the result of leaving the energy partially drained but the device may be capable of continued service, albeit for a shortened period of time.

When discussing denial of energy attacks it is assumed that the device is not plugged in to a constant supply of energy. If it was it would be necessary to have the device consume energy at a rate faster than it is being supplied to the device and this is not a realistic scenario for an overwhelming majority of devices. For this reason we make the aforementioned assumption.

5.1.1 Denial of Service

Inspired by the identity based DoS attacks in Bellardo and Savage[4] is this "reverse identity attack" where the attacker will spoof the AP instead of the victim. In doing so the attacker can now send control and management frames to the victim in an attempt to elicit certain behavior. For example, the attacker could force the victim to stay awake by saying that the AP does not support the sleeping functionality. The attacker could also continuously tell the victim that it has data available, causing it to continuously request this phantom data from the AP. Important in both of these examples is that the victim is sleeping far less than in normal operation. By reducing sleep time one can greatly increase energy used since wireless chipsets use up to 90% less power while sleeping[13].

The most effective defense for such attacks would be MAC level authentication, such as that provided by IPSec or the upcoming 802.11i standard . Unfortunately it does not appear that other short range wireless protocols are taking the same precautions.

The plausibility of such an attack is very high, especially considering the highly pervasive nature of 802.11 networks in today's society. The effectiveness will require further study as there have not been many papers regarding energy abuses of wireless protocols.

5.1.2 Open Services

Open Services are services on a device which are generally not protected and offer access to data or specialized hardware the device might offer. One example is iTunes Music Sharing, a service provided by the application iTunes which allows devices to connect to a shared music library and listen to its contents. Each time a remote client accesses a song it must be accessed on disk then streamed over the network, all of this has a very high energy cost. These services can be exploited anonymously by the public at large due to their open nature. Open services can be provided by the host operating system or applications running on top of it.

Here is a short, and by no means complete, list of open services which could be exploited.

- Bluetooth keyboards/mice
- ssh, ftp, httpd, vnc, dns
- DAAP (via iTunes Music Sharing)

- Zune music sharing (proprietary)
- GPS navigation devices
- Zeroconf (a service for publishing services)
- distributed compilation
- svn and cvs (versioning systems)
- rpc (remote procedure calls)

It is important to note that several of these open services are guarded by proprietary protocols or weak forms of authentication, these are negligible obstacles to a resourceful hacker. I believe that open services present one of the greatest vulnerabilities for energy attacks since they often leverage high-cost internal functionality of devices. That is, they offer up information which requires heavy computation, data retrieval, or both. In some instances the services on a device are not designed specifically for that device, rather they are generally designed to work on many devices. In these cases it is probable that the service does not take energy efficiency into account and is that more vulnerable.

In section 6 I describe one specific exploit based on iTunes Music Sharing and propose experimentation based on it. The exploit is relatively simple and can be used as denial of energy attack when applied to a majority of laptops in the market. Furthermore it could also be used as a cost of energy attack if applied to a desktop machine.

The plausibility of attacking open services is very high. Several of these applications provide data and computation rich services which are energy intense. Furthermore as applications move forward we are seeing an increasing trend towards providing more application services such that applications can integrate with each other. While this trend will foster innovation it leaves the device open to energy attacks.

5.1.3 Closed Services

Closed Services are services which guard a device such as some sort of secure authentication or a firewall. To determine if these systems are vulnerable to energy attacks a certain amount of experimentation about their energy usage must be done. To explore what questions must be answered let us consider a firewall. Do firewalls keep all of their rules in memory or do the have to go to disk to access infrequently used rules? Do they differentiate between "more secure" and "less secure" ports that is, is the same function used on every port to determine if packets should be rejected or denied? What is the energy cost of rejecting a packet? What is the energy cost of rejecting a stream of similar packets? and so on.

Over a short interval closed services would be difficult to attack and would likely not be worth it. There may, however, be a payoff over a long attack.

5.1.4 Bluetooth

Bluetooth is a common short range peer to peer protocol for file and information exchange which is often included in laptops, cell phones, and handheld computers. There are many known vulnerabilities in bluetooth[21] which can be currently exploited to gather sensitive information off devices with out a user's knowledge. Additionally there exist methods to forcibly push information to devices. Each time this happens the device is forced to use its bluetooth chipset and store data on its long term storage. In most devices the Bluetooth chipset incurs little cost when idle and a large cost when active. This can be exploited by simply sending requests which do not require authentication or pairing, i.e. requests that can be done without requiring the interaction of the person using the victim device. For example a series of discovery requests, information requests, or pairing requests would do this.

Bluetooth related attacks provide a very large opportunity for energy attacks with one major problem: it is a *short range* protocol. The effective range of a Bluetooth devices is less than 30 feet. This means the attacker is either in the same room as the victim or the attacks are spread via a virus. It is important

to note that bluetooth viruses are difficult to implement due to the diversity of operating systems found in devices which support bluetooth.

To guard against such attacks devices could limit the number of any one kind of request over a given time period. This would limit the attacker's ability to perform the attacks and could be easily implemented in an embedded system.

5.1.5 Cellular

Cellular chipsets are becoming increasingly prevalent in mobile devices. Aside from cell phones they are also found in some laptops and the Amazon Kindle[1]. Many high end cellular devices also include Bluetooth and 802.11 as well. This makes them ripe targets for energy attacks.

Most cellular devices can communicate using the SMS and MMS protocols, or Short Message Service and Multimedia Message Service respectively. This is a service available on nearly all cell phones which can be used to perform a denial of energy attack with staggering economic consequences for the victim. Cell phones use dramatically less energy when not actively communicating with a cellular tower. By sending SMS or MMS messages to a device it must activate its cellular chipset. The device must also go to its persistent storage to store the data; this is usually flash storage which does not require much energy to access, but does require some energy none the less. In addition to any energy cost incurred a fiscal cost is also incurred each time a SMS or MMS is sent or received. Unless the victim has an "unlimited[2]" plan it will cost them between 10¢ and $2 per message. Also important is that most cellular providers allow e-mail to SMS communication, in these cases there is only a charge associated with the cellular side of the communication.

This technique was recently explored by Racic, Ma, and Chen[16]. In their paper they exploit vulnerabilities in MMS and propose a two phase system to exploit them. The first stage involves creating a "hit-list" of devices which are vulnerable. This process is essentially phishing, where they send out several MMS notification messages containing an address to a malicious web server. They know that all of the devices that "bite" by responding to the request are vulnerable. Using this technique they were able to drain a cellular phone's battery 22 times faster than were it left idle.

This attack is one of two energy attacks which has been successfully demonstrated and, as stated above, is quite efficient.

5.1.6 Location Services: GPS and Skyhook

GPS based navigation systems, such as those produced by Garmin and TomTom, have become popular recently. GPS functionality is also included in several higher end cell phones and I believe will soon be standard on nearly all mobile devices, this especially includes laptops and hand held computers, cell phones, and cameras. While *very* difficult to hack into a satellite or do any sort of identity based attack (i.e. spoof a satellite) these devices do sometimes offer up services which other devices can utilize. There may be some nature of attack if you could alter the GPS data received. For instance a device may run a "sanity check" function when receiving data which exceeds some bounds relative to the previous received information. If this is the case, then it represents an opportunity for exploitation.

This attack is amongst the least plausible. For starters it is difficult to hack a satellite's signal. If you did manage to hack into the signal then NASA and the FBI will be wanting to talk with you. On the device's side of things GPS receivers are designed to poll the satellites at regular intervals, so they are already operating at a higher energy usage.

Generally GPS chipsets are large and power hungary (for reasons stated above), so contemporary devices such as cell phones and 802.11 devices can use triangulation services such as that offered by Skyhook. These services use triangulation from cellular towers combined with information given by WiFi access points to get a very accurate position. This kind of functionality is available on both the iPhone and Android operating systems. Since it relies on access point data it is significantly easier to attack than a pure GPS system. By spoofing an access point and giving information that contradicts the cellular towers the location system

[1]The Kindle is a device for reading e-books.

[2]Unlimited plans are usually limited to a large number, so this is for values of unlimited on the order of 1000.

could be forced to continually re-check its position with the cellular towers. This would force undue amounts of cellular communication and would drastically effect the device's battery life.

5.2 Wired Attacks

Where the goal of the wireless attacks was to run down an energy source the goal of wired attacks are fundamentally different. This is mostly since wired devices don't have limited energy supplies. The goal of a wired energy attack is not to temporarily deny service, rather it is to run up the cost of operating the victim or to physically destroy it. The latter option is rapidly disappearing as modern processors have temperature based kill switches, however, this technology is not as common in RAM, hard disk drives or other auxiliary processors on the motherboard. We can expect sensor technology to continue to spread, however, over the next several years.

5.2.1 Single Machine

Energy cost maximization can be performed against a single machine using similar techniques to the wireless attacks, I believe there are ways to physically damage machines but more research is required before making any definitive statements.

5.2.2 Server Farm

Server farms, such as those used for search engines or scientific computing, have enormous energy requirements when at full load. The computers and networking components produce so much heat they require specialized cooling systems. These cooling systems also require large amounts of energy. When attacking a cluster you will cause more cooling to engage, this means the attack now has a greater effect. Clusters which provide open services, such as searching or allow public access, are vulnerable to cost of energy attacks.

To exploit open services on clusters the goal it to *maximize the per transaction cost*. In other words we want to craft queries which make the cluster do more computation than normal. The question is how do we know when a cluster is doing more computation? Search engines suffer the flaw of hubris, they like to brag about how quickly they were able to generate your search results page. This information can be used against them since it relates to the amount of computation required to generate your page.

Using a dictionary based attack one could produce an algorithm that would find search terms which took longer times to generate pages for. Why does this work? A majority of this time is spent performing the MapReduce[5] operation over a distributed file system[9]. The longer this algorithm takes the more energy is used in generating its output. This problem seems suited to genetic algorithms which can auto-tune by breeding algorithms.

For clusters which provide other services such as e-mail, media streaming, or file hosting other attacks are possible. These can be attacked in manners similar to the wireless open services. Unlike the wireless services, however, to run a cost effective attack a much larger attack base is required. This could be another server farm or better yet a bot net.

To guard against such attacks is quite difficult. How do you differentiate between legitimate queries and malicious ones? Google already filters out some search spam, but their filters rely on misspellings and won't catch a group of five esoteric words. Suppose a large bot net is performing the above attack, the best defensive strategy would be a behavior based filter. One could block searches that have a rapid and regular period for instance.

A more steadfast defense is to make your energy cost less by using renewable sources such as solar. Doing so, however, introduces batteries which are vulnerable to attack. In this scenario the strategy would be to attack repeatedly at night since batteries only recharge a set number of times after a year of attacks your batteries are all dead. While this is a very long term attack, it still is worth considering.

5.3 Propagation

Depending on the intent of the attacker a large variety of options are available for distribution and propagation of the attacks. A worm would be amongst the most effective, allowing a device to attack other devices when they are in range and also perform local attacks when there are no other possible victims. Also available is a bluetooth virus. There are currently several viruses that spread via bluetooth cell phones and laptops, there is even one which spreads by way of a cars keyless entry system.

When considering wired attacks a botnet is a suitable vehicle. When attacking a large data center an equally large attacking force is probably needed, a botnet can provide this kind of attack base. Additionally the distributed nature of a botnet makes the attack far more difficult to detect and it also lessens the risk of discovery for the attacker.

6 Experimentation

6.1 Method

The experiment focused on the open services model. Each attack involved only two machines, one attacker and one victim. The only requirements are 802.11 and the victim must be battery powered. The victim runs a suite of common applications with open services. This suite includes ssh, ftp, ping, and iTunes. The only special application configuration required is that iTunes music sharing is enabled. Also on the victim is a cron script which runs once every minute to sample the battery status[3]. To better gauge the effectiveness of the attacks all unused peripherals were turned off, the monitor backlight was dimmed, and cpu scaling was turned off.

The attacker system is similarly equipped to the victim. It is running the same suite of applications with a set of scripts to automatically run a series of attacks on the victim.

6.2 Results

To discuss the effectiveness of an energy attack one needs to first know some other information about the victim device. First you must know the victim's "normal" operational time, these can vary according to measurements so the idle operational time also works. The attack will aim to reduce this number as much as possible. Also useful is to know the lower bound, for energy attacks this is the device's minimum operational time (though this is also difficult to obtain). This lower bound is the best an attack can do, so it is the attack's target.

The first attack is a straightforward media streaming attack. Each file is streamed for five seconds before switching to the next. The media begins streaming nearly immediately. iTunes works like most streaming servers where it sends the entire file over to the client, so this attack could be enhanced by dynamically detecting when the entire file has been streamed and skip to the next. The results for attack two are shown in figure 6.

To improve upon the attack the media is switched to movie files only. Each file is streamed for 15 seconds before switching to the next. The time has been increased to cover the increased overhead of loading a movie file on the victim. When the attack was performed with a period of five seconds the media never actually started streaming. The results for attack two are shown in figure 7.

For more detail on what is actually going on figure 2 shows the number of amperes discharged each minute for an idle machine. Figure 3 and figure 4 show each minute of the attack for scenarios one and two respectively. In figure 4 there is a large spike in the last two minutes, this is because the machine suspended the streaming to go into "safe sleep" mode. This is a mode where the ram is backed up to the hard disk when the computer is low on power.

[3]Script available in section 10.2.

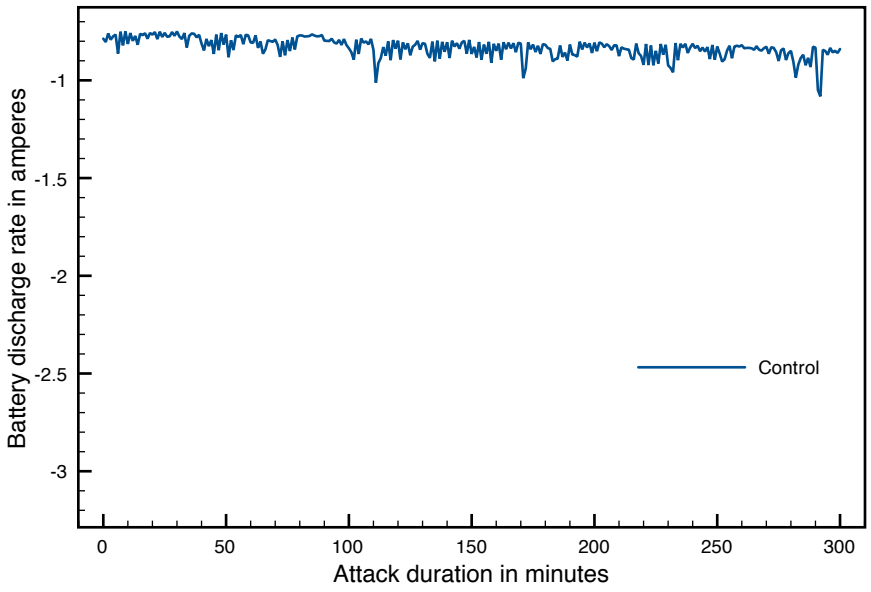

Figure 2: Battery discharge over time for an idle laptop.

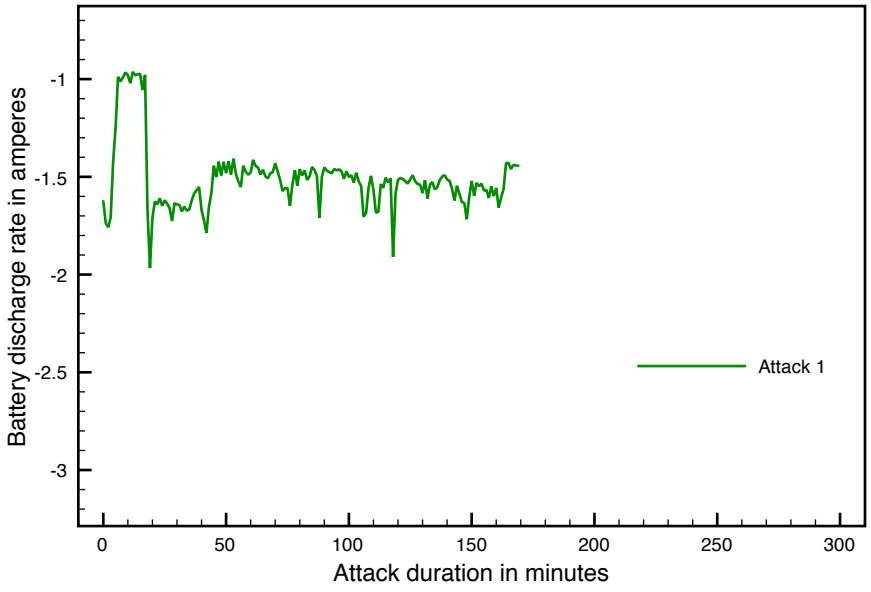

Figure 3: Battery discharge over time for attack one.

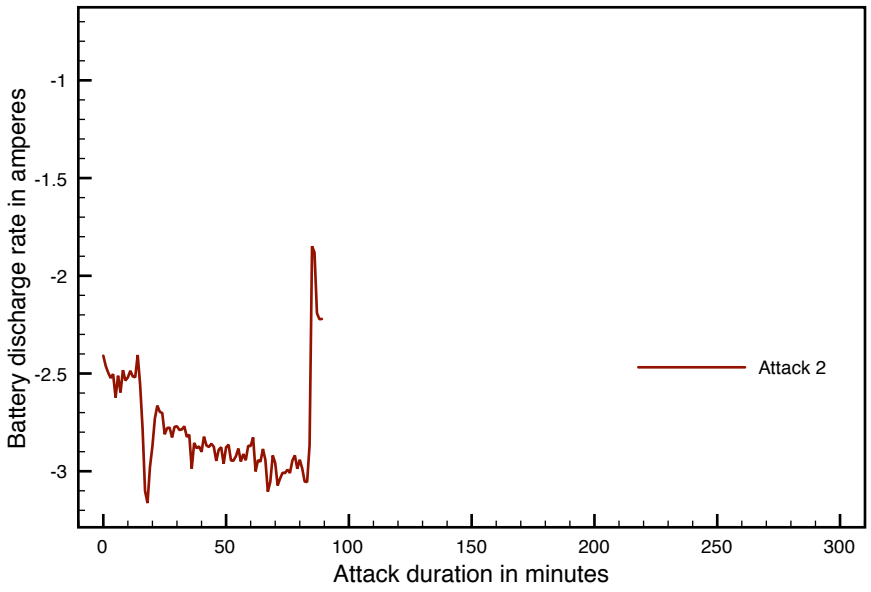

Figure 4: Battery discharge over time for attack two.

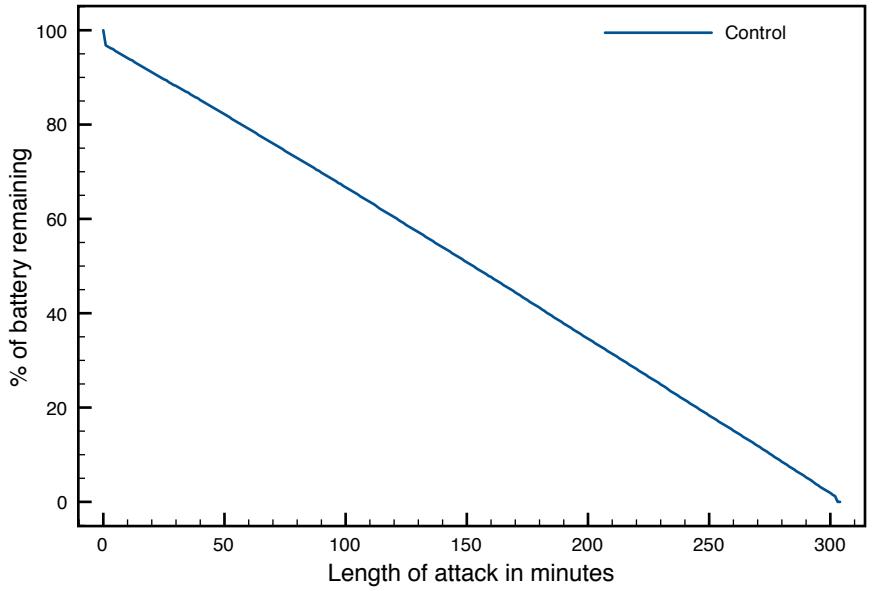

Figure 5: The control case of an idle laptop.

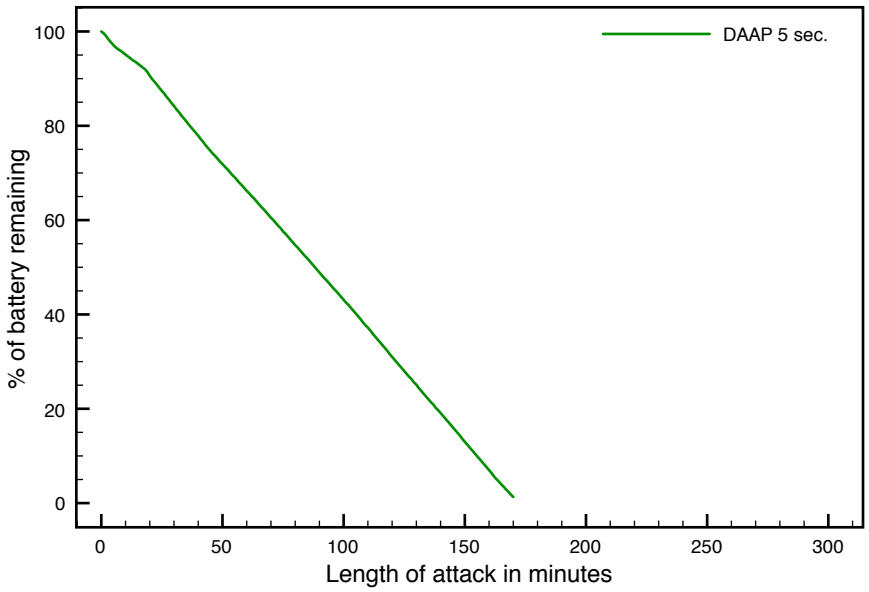

Figure 6: Attack scenario 1: streaming music files for five seconds each.

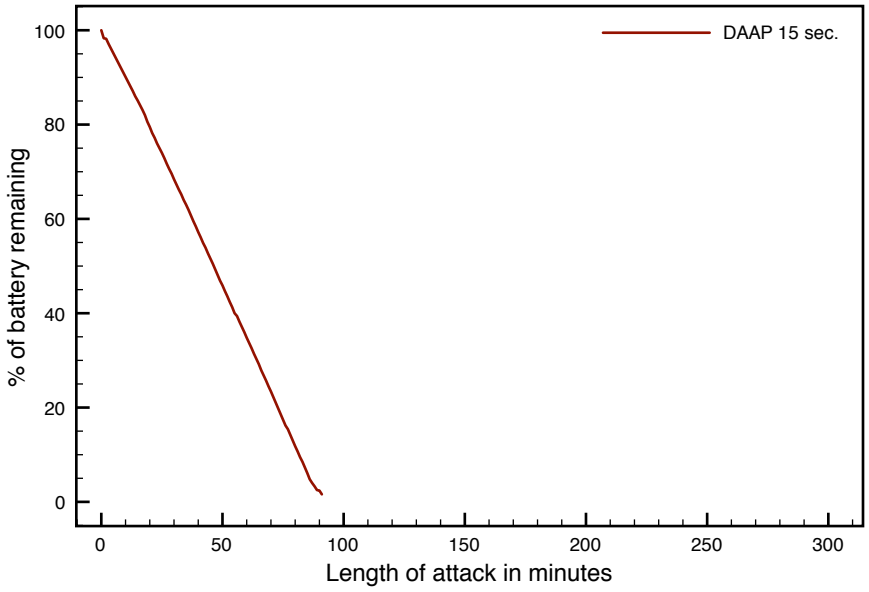

Figure 7: Attack scenario 2: streaming movie files for five seconds each.

7 Future Work

Server farms, such as those used for search engines or scientific computing, have enormous energy requirements when at full load. The computers and networking components produce so much heat they require specialized cooling systems. These cooling systems also require large amounts of energy. When attacking a cluster you will cause more cooling to engage, this means the attack now has a greater effect. Clusters which provide open services, such as searching or allow public access, are vulnerable to cost of energy attacks.

To test a cost of energy attack I will build a small cluster and install a search engine on it. The search engine will be based Hadoop[4][1], an open source implementation of Google's MapReduce[5] and Goggle File System[9]. This search engine will be available by way of a web page, giving us an open service to exploit.

In other words we want to craft queries which make the cluster do more computation than normal. The question is how do we know when a cluster is doing more computation? Search engines suffer the flaw of hubris, they like to brag about how quickly they were able to generate your search results page. This information can be used against them since it relates to the amount of computation required to generate your page. Our goal will be to increase the per-transaction computation, where a transaction is a singe search query. The attacks will have the goal of increasing the per-transaction cost by finding higher energy cost search queries. This will be done using a dictionary based attack to find search queries which take longer to return. A longer search time means more work had to be done in MapReduce this translates to more computation and thus higher energy consumption.

Using a dictionary based attack one could produce an algorithm that would find search terms which took longer times to generate pages for. Why does this work? A majority of this time is spent performing the MapReduce operation over a distributed file system. The longer this algorithm takes the more energy is used in generating its output. This problem seems suited to genetic algorithms which can auto-tune the attack as the victim caches information by breeding algorithms.

The following measurements will be taken:

- Idle workload
- "Normal" workload
- "Heavy" workload
- Attack workload

If time permits the test should be run on two different clusters each with different hardware configurations. The details of these configurations is listed in table 1. While the hardware may seem dated this should not be considered a problem. Since we are looking at the difference in cost between an *average* transaction and a *malicious* transaction the results should still apply to clusters with different configurations.

	Tornado: Cluster of 32 Nodes	Hurricane: Cluster of 4 Nodes
Model	Sun Ultra 60 Model 2360	Sun Enterprise 420R
Processor	2x UltraSPARC II 360 MHz 4 MB L2 cache	4x UltraSPARC II 450 MHz 4 MB L2 cache
Memory	512 MB memory	4 GB memory
Storage	18.2 GB 10000 rpm UltraWide SCSI	18.2 GB 10000 rpm UltraWide SCSI
Network	1280 Mb/s Myrinet	1000 Mb/s Ethernet and 2000 Mb/s Myrinet
Power	Single power supply	Dual power supplies

Table 1: Hardware configurations of the two testbeds I will be testing on.

[4]Hadoop is available at `http://lucene.apache.org/hadoop/`.

8 Related Work

While there is a lot of work being done on energy efficiency there is very little on energy related security issues. Bellardo and Savage[4] give a mention to an identity based DoS attack which utilizes the energy saving mechanisms in 802.11 chipsets. By masquerading as the victim whilst it sleeps they are able to fool the AP into prematurely sending any of the victims queued data. When the victim wakes from its slumber it finds no data. This technique has been proven to be very effective at interrupting communication.

This area has also recently been explored by Racic, Ma, and Chen[16]. In their paper they propose using MMS messages to drain a cellular phone's battery 22 times faster than "normal use". They never reference where their normal use numbers come from. The manufacturers websites only reference the information provided in table 2. Regardless their results are still impressive. They were able to come close to the maximum talk times of the devices, and in two cases get beneath it. These results, however, can not be easily compared to other kinds of mobile devices — such as smart phones, laptops, and UMPCs. This is because cellular phones have idle operation times on the order weeks whereas smart phones and laptops have idle operation times on the order of hours to days.

Phone	Talk Time		Standby Time		Attack Time
Nokia 6620	4 hours	(240 minutes)	200 hours	(8.3 days)	7 hours
Sony-Ericsson T610	14 hours	(840 minutes)	315 hours	(13.1 days)	7 hours
Motorola v710	3.3 hours	(200 minutes)	150 hours	(6.3 days)	2* hours

Table 2: Data on phones provided by manufactures compared to their attack times. *Note that the attack here was using the phone as a wireless modem, this will add considerable strain to the phone and its operational time as a modem differs from its operational time as a phone.

9 Conclusion

In this paper I have presented several thoughts on the new field of energy security through the discussion of many ways to attack the energy supplies of mobile and wireless devices as well as servers and large scale data centers. Through experimentation I have shown the threat is real by reducing a normally six hour battery down to one and a half hours. I believe the field of energy security will grow and continue to become increasingly important over the next several years.

10 Apendix

10.1 iTunes Script

```
#!/usr/bin/ruby
require 'osx/cocoa'
include OSX
require_framework 'ScriptingBridge'
iTunes = SBApplication.applicationWithBundleIdentifier:'com.apple.iTunes'
#iTunes.playpause

while true
  iTunes.nextTrack
  puts "Now playing: #{iTunes.currentTrack.name}"
  sleep 5
end

puts "Done"
```

10.2 Battery Script

```
#!/bin/zsh

IOREG="/usr/sbin/ioreg"
VERSION="1.4"

#
# check_for_ioreg
# - Check that the ioreg program is available.
#
# input:    nothing
#
# output:   nothing
#           return code     - status
#
check_for_ioreg ()
{
    if [ ! -x $IOREG ]; then
        echo "battery: can not execute $IOREG" >&2
        exit 1
    fi

    return 0
}

#
# get_battery_info
# - Get battery information.
#
# input:    nothing
#
# output:   stdout          - battery info
#           return code     - status
#
get_battery_info ()
{
    local line
    local line1
    local line2

    $IOREG -p IODeviceTree -n "battery" -w 0 | grep IOBatteryInfo | {
        read line
        line1=${line:s/IOBatteryInfo/ BATTERY 1 /}
        line2=${line1:s/\}\,\{/ BATTERY 2 /}
        echo "${line2//[|\"=\(\{\}\),]/ }"
    }

    return 0
}

display_battery_info ()
{
    local how=$1
    local line
    local name
    local value
    local battery
    local voltage1 flags1 amperage1 capacity1 current1 cyclecount1 absolutemaxcapacity1
    local voltage2 flags2 amperage2 capacity2 current2 cyclecount2 absolutemaxcapacity2

    read line
    echo ${=line} | sed 's/Cycle Count/CycleCount/g' | xargs -n 2 echo | while read name
        value; do
```

```
        case ${name:l} in
        battery)
            battery=$value
            ;;
        voltage|flags|amperage|capacity|current)
            eval ${name:l}$battery=$value
            ;;
        cyclecount|absolutemaxcapacity)
            eval ${name:l}$battery=$value
            ;;
        esac
    done

    amperage1=$(expr $amperage1 + 0)                # Tiger fix
    [ ! -z "$voltage2" ] && amperage2=$(expr $amperage2 + 0)    # Tiger fix
    display_one_battery_$how 1 $voltage1 $flags1 $amperage1 $capacity1 $current1
        $cyclecount1 $absolutemaxcapacity1
    [ ! -z "$voltage2" ] && display_one_battery_$how 2 $voltage2 $flags2 $amperage2
        $capacity2 $current2 $cyclecount2 $absolutemaxcapacity2

    return 0
}

#
# display_one_battery_compact
# - Display information about one battery, compact form.
#
# input:    $1          - battery number
#           $2          - voltage in mV
#           $3          - flags
#           $4          - amperage in mA
#           $5          - max capacity in mAh
#           $6          - current capacity in mAh
#           $7          - number of discharge cycles
#           $8          - absolute max capacity for battery in mAh
#
# output:   stdout      - battery info in compact form
#           return code - status
#
display_one_battery_compact ()
{
    typeset -i battery=$1
    typeset -F 3 voltage=$2
    typeset -i 2 flag_byte=$3
    typeset -F 3 amperage=$4
    typeset -F 3 capacity=$5
    typeset -F 3 current=$6
    typeset -i cycles=$7
    typeset -F 3 maxcapacity=$8

    typeset -Z 16 zero_filled_flag_byte=${flag_byte/2#/}
    typeset flags=$zero_filled_flag_byte
    typeset -F 1 percentage
    typeset -F 1 maxpercent

    #
    # Print battery battery number, flags, voltage, current, and capacity.
    #
    # Convert mV, mA, and mAh to V, A, and Ah.
    # Also calculate the current capacity as a percentage
    # of the full battery capacity.
    #
    (( voltage  /= 1000 ))
    (( amperage /= 1000 ))
    (( capacity /= 1000 ))
```

```
    (( current  /= 1000 ))
    (( maxcapacity /= 1000 ))

    if [[ $capacity != 0.000 ]]; then
        (( percentage = 100 * current / capacity ))
    else
        (( percentage = 0 ))
    fi
    if [[ $maxcapacity != 0.000 ]]; then
        (( maxpercent = 100 * capacity / maxcapacity ))
        print "$(date +'%Y%m%d %H%M%S')"        \
              "$battery"                        \
              "$flags"                          \
              "$voltage $amperage"              \
              "$current $capacity $percentage"  \
              "$maxcapacity $maxpercent"        \
              "$cycles"
    else
        print "$(date +'%Y%m%d %H%M%S')"        \
              "$battery"                        \
              "$flags"                          \
              "$voltage $amperage"              \
              "$current $capacity $percentage"
    fi

    return 0
}

#
# main
# - Display battery info.
#
# input:    $1              - how to display
#
# output:   stdout          - battery info in selected text form
#           return code     - status
#
main ()
{

check_for_ioreg
get_battery_info | display_battery_info compact

return 0

}

#
# Start main.
#
main ${1+$@}; exit $?
```

References

[1] The hadoop project.

[2] Manish Anand, Edmund B. Nightingale, and Jason Flinn. Self-tuning wireless network power management. *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2003.

[3] Manish Anand, Edmund B. Nightingale, and Jason Flinn. Ghosts in the machine: Interfaces for better power management. *Proceedings of the 2nd Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[4] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. *Proceedings of the USENIX Security Symposium*, pages 15–27, August 2003.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI: Sixth Symposium on Operating System Design and Implementation*, 2004.

[6] Jeffrey Dean and Sanjay Ghemawat. *Beautiful Code*. O'Reilly, 1 edition, 2007.

[7] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, December 1999.

[8] Jason Flinn and M. Satyanarayanan. Powerscope: a tool for profiling the energy usage of mobile applications. In *Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 2–10, February 1999.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. *19th ACM Symposium on Operating Systems Principles*, 2003.

[10] Urs Hoelzle and Bill Weihl. High-efficiency power supplies for home computers and servers. *Whitepaper*, 2006.

[11] IEEE 802.11 Working group. *802.11 Standards documents*.

[12] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI)*, 2006.

[13] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. In *MobiCom 2002*, Atlanta, GA, September 2002.

[14] Thomas Moscibroda and Onur Mutlu. Memory performance attacks: Denial of memory service in multi-core systems. February 2007.

[15] Edmund B. Nightingale and Jason Flinn. Energy-efficiency and storage flexibility in the blue file system. *Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[16] R Racic, D Ma, and H Chen. Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery. *Proceedings from the 15th USENIX Security Symposium*.

[17] P. Traynor, P. McDaniel, and T. La Porta. On attack causality in internet-connected cellular networks. *USENIX Security Symposium (SECURITY)*, August 2007.

[18] Patrick Traynor, William Enck, Patrick McDaniel, and Tom La Porta. Mitigation attacks on open functionality in sms-capable cellular networks. *16th USENIX Security Symposium*.

[19] Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson. Secretly monopolizing the CPU without superuser privileges. In *USENIX Security Symposium*, pages 239–256, Boston, Massachusetts, August 2007.

[20] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. DDoS Defense by Offense. In *ACM SIGCOMM 2006*, Pisa, Italy, September 2006.

[21] Wikipedia. Bluetooth — wikipedia, the free encyclopedia, 2007. [Online; accessed 6-December-2007].

[22] Wikipedia. Hadoop — wikipedia, the free encyclopedia, 2007. [Online; accessed 6-December-2007].

[23] Wikipedia. Ieee 802.11 — wikipedia, the free encyclopedia, 2007. [Online; accessed 6-December-2007].

[24] Wikipedia. Northbridge (computing) — wikipedia, the free encyclopedia, 2007. [Online; accessed 6-December-2007].