On Energy Security of Server Systems

Zhenyu Wu, *Member*, *IEEE*, Mengjun Xie, *Member*, *IEEE*, and Haining Wang, *Senior Member*, *IEEE*

Abstract—Power management has become increasingly important for server systems. Numerous techniques have been proposed and developed to optimize server power consumption and achieve energy proportional computing. However, the security perspective of server power management has not yet been studied. In this paper, we investigate energy attacks, a new type of malicious exploits on server systems. Targeted solely at abusing server power consumption, energy attacks exhibit very different attacking behaviors and cause very different victim symptoms from conventional cyberspace attacks. First, we unveil that today's server systems with improved power saving technologies are more vulnerable to energy attacks. Then, we demonstrate a realistic energy attack on a stand-alone server system in three steps: 1) by profiling energy cost of an open web service under different operation conditions, we identify the vulnerabilities that subject a server to energy attacks; 2) exploiting the discovered attack vectors, we design an energy attack that can be launched anonymously from remote; and 3) we execute the attack and measure the extent of its damage in a systematic manner. Finally, we highlight the challenges in defending against energy attacks, and we propose an effective defense scheme to meet the challenges and evaluate its effectiveness.

Index Terms—Energy attack, server security, energy-aware programming

1 Introduction

Power management is one of the critical issues for server systems nowadays. To date energy cost has become a major factor in the total cost of ownership (TCO) of large-scale server clusters [3], [17]. According to EPA [34], more than 100 billion kilowatt hours, representing a $7.4 billion annual cost, are estimated to be consumed by data centers in US by 2011. As the price of hardware keeps dropping while its performance continuously improves, the proportion of energy cost in overall expense of server systems tends to grow even larger [3], [17].

Previous research on server power management mainly focuses on reducing power consumption while maintaining acceptable quality of service. Numerous techniques have been proposed to improve energy efficiency in a variety of aspects, from low-level hardware features such as processor Dynamic Voltage and Frequency Scaling (DVFS) [13], [19] and hard disk spin-down [8], [16], to high-level system-wise management schemes such as cluster load provisioning [9], [27] and virtual machine consolidation [25]. While these power management advancements have significantly improved power savings, they have also opened up spaces for energy misuse. However, the security aspect of server system power management has not yet been paid attention to.

In this paper, we investigate energy attacks, a new type of malicious exploits on server systems. Stealthily launched from remote by anonymous attackers, energy attacks increase the power consumption of a server system nonproportionally to its effective workload. Energy attacks are distinct from conventional cyberspace attacks in three interrelated aspects: objectives, attacking behaviors, and victim symptoms. First, energy attacks aim solely at abusing power consumption. They do not attempt to disrupt the normal service operations of the victim servers, nor to acquire sensitive information from the victim servers. Second, energy attacks are mounted in a stealthy manner, and they deliver damages over a relatively long period of time. An attacker's network flow is indistinguishable from those of the normal clients, in terms of traffic patterns or data fingerprints. Third, energy attacks manifest on victim servers only as increased energy usage, and no other induced anomalies such as significant performance degradation.

Although no immediately observable damages ensue, the consequences of energy attacks are serious. A successfully launched energy attack can cause a victim system to waste a large amount of energy, which in turn becomes waste heat, resulting in significantly increased power and cooling expense, shortened hardware component lifespan, reduced reliability, and sometimes even permanent hardware failure. Current power management and security mechanisms provide virtually no defense against energy attacks.

To demonstrate the feasibility of launching an energy attack, we perform a step-by-step design and execution of a realistic energy attack on Wikipedia mirror server. First, we profile the power consumption of the victim web server under different page serving conditions, and identify a condition that triggers high energy consumption as a viable attack vector. We then proceed to design an energy attack technique, achieving stealthiness by leveraging knowledge of human web browsing behaviors. Finally, we evaluate our design by executing the attack and systematically measure the power consumption increases of the victim server under different load conditions. We observe that the effect of the energy attack is dependent on the existing workload of the

- *Z. Wu is with NEC Laboratories America, Inc., 4 Independence Way, Suite 200, Princeton, NJ 08540. E-mail: adamwu@nec-labs.com.*
- *M. Xie is with the Department of Computer Science, University of Arkansas at Little Rock, 2801 South University, Little Rock, AR 72204. E-mail: mxxie@ualr.com.*
- *H. Wang is with the Department of Computer Science, College of William and Mary, PO Box 8795, Williamsburg, VA 23187. E-mail: hnw@cs.wm.edu.*

Manuscript received 25 Aug. 2011; revised 28 Mar. 2012; accepted 24 July 2012; published online 2 Aug. 2012.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2011-08-0199. Digital Object Identifier no. 10.1109/TDSC.2012.70.

victim server system. And at typical workloads, our attack is able to increase a victim server's power consumption from 21.7 to 42.3 percent.

In seeking of an effective defense against energy attacks, we realize that the current server hardware lacks support for fine-grained power measurement, a critical component for building general purpose defense system. We propose an application-oriented defense scheme utilizing energy-aware programming to work around the missing hardware support. We show a proof-of-concept implementation, and evaluate its effectiveness in defending against an energy attack.

The remainder of this paper is structured as follows: Section 2 presents the background on server system energy saving and the security implication. Section 3 details the design of energy attacks. Section 4 evaluates the threat of the proposed energy attack. Section 5 presents our software-based defense scheme. Section 6 further discusses related issues. Section 7 surveys related work on power management and energy security for server systems and mobile devices. Finally, Section 8 concludes the paper.

2 Background

In this section, we first discuss the impact of energy proportional computing on a server system and present power measurements on our own server systems. Then, we describe the threat of energy attacks exposed on today's server systems.

2.1 Power Distribution

The power consumption in a server is mainly attributed to two sources, system component powering and cooling, with the latter heavily dependent on the former. Server system components mainly fall into the following categories: power supply, motherboard (chipset), processor, memory, and disk storage.

The power supply is responsible for transforming high voltage electricity input from a power outlet to a proper form of electricity (e.g., 5 V, 12 V DC) for all other server components. Although the power supply does not directly participate in service processing, it consumes a portion of input power due to conversion loss. The state-of-the-art power supplies guarantee over 90 percent efficiency at normal loads (i.e., 20-100 percent of rated output) [35].

The motherboard and chipset provide the basic interconnection for all other system components. Modern server system chipset has nominal TDP[1] from 25 to 35 W. Processors are usually the component that is capable of consuming the most of power per unit. A typical server processor's TDP is rated at 65 to 130 W, and a server is usually equipped with one to four processors. Memory, typical Fully Buffered DRAM (FBDIMM) for servers, has per-unit TDP of approximately 12 W [20]. A server system usually has four to eight memory modules installed, which add up the overall memory power consumption to 48 to 96 W.

The disk storage for a server system usually consists of multiple hard drives. The power consumption of a hard drive is mainly determined by its disk rotation speed: 7,200 and 10,000 Rotations Per Minute (RPM) drives consume 8 W power while idling and 12 W on average [32], but 15,000 RPM drives consume 12 W idle and 16 W on average [33].

1. Thermal Design Power, a reference number of the typical amount of power a processor or chipset draws during full utilization, which is usually lower than its peak power consumption.

2.2 Energy Proportionality

Energy proportional computing [4] is an important concept in today's server systems. It aims to address the increasing energy concern and demand for power saving by making servers consume energy proportional to its workload. This goal is normally achieved by conditionally trading off performance for power savings.

Processors are the primary targets for power optimization, because of their high maximum power consumption (hundreds of watts per unit). Nowadays, the majority of server-class CPUs have employed power saving techniques that are already used in desktop and mobile processors, such as DVFS, multiple power states with reduced performance, and even power-down of idle cores. Motherboard and chipset feature the shutdown of unused circuitry, and memory chips also have several standby states with reduced power for no read/write cycles. Hard drives can only save a small portion of energy at idleness, due to their power demanding internal mechanical parts (spinning platters). However, they have another power saving mechanism called "spin-down," which shuts down the motor and thereby cutting down the majority of the power consumption, at a high (latency) cost of resuming service.

The Advanced Configuration and Power Interface (ACPI) specifications [1] are introduced to unify the power management of various types of devices in computer systems and provide well-defined power management interfaces for both hardware and software. Within the specifications, multiple performance states are defined for a computer component. Each performance state corresponds to a specification of the expected performance and power consumption. At least one state is well defined: a full power state corresponds to the maximum performance. Depending on device type and manufacturing technology, additional number of reduced performance states can be defined.

Although modern operating systems are all capable of utilizing the ACPI to conserve energy under light load or in idleness, previous generations of server systems (such as our System A below) are not very energy proportional. This is because performance and security used to be the primary concerns for server systems, and thus the underlying hardware provides little or no support of additional performance states with reduced power consumption. However, as energy concerns weigh increasingly heavily, today's server systems have been becoming more energy proportional.

2.3 Real Server Measurements

We perform a small measurement study on system power consumption, using two server systems with different generations of hardware configurations, which are listed in Table 1. System A was bought in 2006 and System B was bought in mid-2009. We believe that both servers are representative of the mainstream system configurations at the time of purchase.

We measure the whole system power consumption in three different load scenarios: completely idle (IDLE),

TABLE 1
Configurations for Server Systems

	System A	System B
CPU	2 * Xeon 5130 Dual Core	2 * Xeon 5520 Quad Core
Memory	4 * 1GB DDR2 FBDIMM	6 * 1GB DDR3 FBDIMM
HDD	4 * 7200RPM SATA	6 * 7200RPM SATA

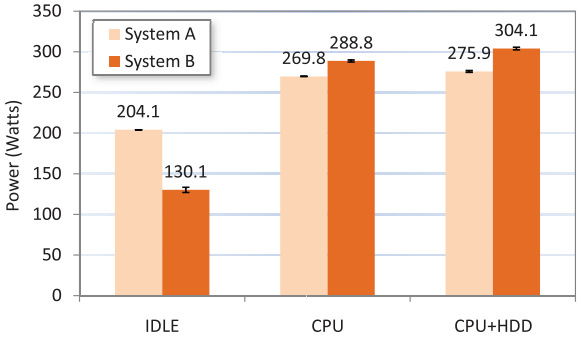


Fig. 1. System power consumptions.

processors being fully utilized (CPU), and processors and hard drives being fully utilized (CPU+HDD). The "CPU" workload is generated by running multiple instances of a classic CPU benchmark program "linpack," and the number of instances corresponds to the number of logic cores. The "CPU+HDD" workload is generated by running the "CPU" workload with the highest `nice` value and, at the same time, writing a large volume of data to the hard drives using the `dd` utility. The power consumption data are collected using a "Watts up? .Net" digital power meter [37], which is capable of measuring power usage with accuracy of $\pm 1.5\%$, at a time granularity of one second.

Two observations can be made from our measurement results shown in Fig. 1: first, in high utilization scenarios System B (the newer server) consumes slightly more power than System A; second, and more interestingly, in the IDLE scenario, the power consumption of System B is significantly less than that of System A. While the first observation can be explained by System B having increased overall computation power than System A, the second observation presents us the direct proof that newer server system is becoming more energy proportional than previous generations. With higher computation power and improved energy proportionality, one can expect System B to yield more energy saving than System A under the same workload. However, we make an additional, alarming observation when we look at the advancements in energy proportional computing from a security perspective.

2.4 Threat of Energy Attacks

The improved energy proportionality has significantly changed the power profile of today's server systems. For example, our measurement data in Fig. 1 shows that compared with IDLE, the CPU+HDD power consumption of System A increases by only 35 percent, while that of System B increases by 134 percent. The larger power consumption increase of System B indicates that it has a wider dynamic power range than System A. In other words, the power consumption of System B (energy proportional server) is more alterable than that of System A (nonenergy proportional server). And the increased power consumption alterability represents a new threat to server systems. The power management mechanism of a server can be attacked by maliciously crafted workloads that target at consuming disproportional amount of energy, rendering the power saving ineffective, and resulting in significant energy wastage of the victim server.

Alarmingly, we realize that the threat of energy attacks is in fact an exploitable vulnerability because currently there is no effective defense against it. Existing power management schemes mainly focus on improving energy efficiency under normal operating conditions with benign workload, and thus they do not provide any defense against energy attacks. Moreover, most server systems do no have an efficient mechanism to measure power consumption, and thus could not even detect energy attacks, let alone defend against them.

2.5 Feature of Energy Attacks

Energy attacks on server systems target at exploiting the aforementioned power management vulnerability, and increasing a victim servers power consumption disproportional to its effective workload. Compared to other cyberattacks, the damage of increased power consumption is delivered in an accumulative fashion over a relatively long period of time. As a result, energy attacks must meet two stealthiness requirements—low network-level signature and low performance degradation.

First, the attack should not exhibit traffic anomalies or have unique traffic patterns, because network traffic is often monitored for security purposes. Second, the attack should cause minimal performance impact on the victim server, as unusual performance degradation is a very visible sign that the server is under attack. The first requirement precludes high service request rate attacks, due to their obvious traffic anomalies. The malicious requests in an energy attack need to be sent at low to normal rate, and hence should be crafted to ensure a high per-request energy cost. In order to fulfill the second requirement, energy attacks must be adaptive to the workload condition of the victim server. Because the victim hosts an open service, its normal workload tends to vary significantly in time (e.g., correlated with the diurnal and weekly cycles). Inflicting a fixed malicious workload on the victim may either result in performance anomaly during high-load periods, or fail to incur the maximum damage during low-load or idle periods.

Note that energy attacks on server systems belong to a new attack class, which is very different from Denial-of-Service (DoS) attacks [21], [31], [36] in terms of their purpose, methodology and effects. Energy attacks aim to stealthily abuse a victim server's power consumption, and try hard to avoid causing any tangible service irregularities. In contrast, DoS attacks target at complete disruption of the victims service, leveraging relatively simple attack strategies such as request flooding. Moreover, because old generations of server systems are not energy proportional, to date

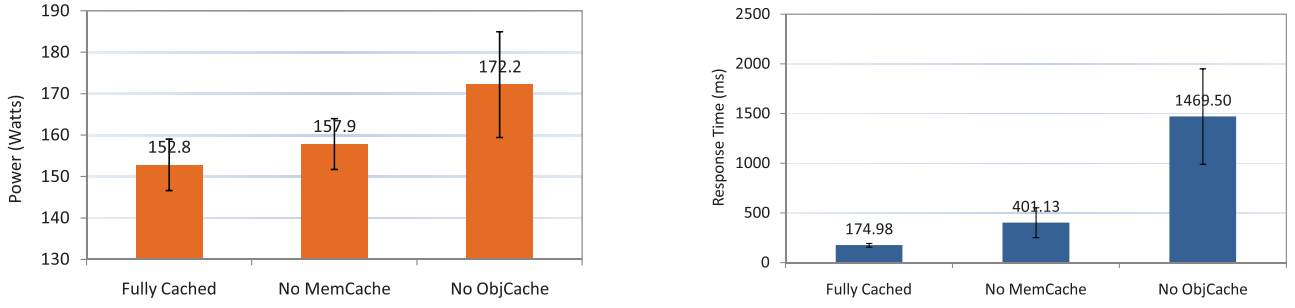


Fig. 2. Power consumption and response time under different caching scenarios.

energy has never been a target for DoS attacks mounted on server systems. And consequently, DoS attacks have mixed energy effects. In other words, not all DoS attacks result in increased power consumption of the victim server, and some could even lead to power consumption decrease. As an intuitive example, a TCP SYN flooding DoS attack exhausts the victim server's socket resource, and thus prevents the victim from receiving normal service requests. This attack causes the major components (e.g., CPUs and hard drives) of the victim server to become idle, and hence significantly reduces its power consumption.

3 Energy Attack on Server Systems

In this section, we demonstrate the feasibility of launching an energy attack. First, we describe the scenario selection. We then design a realistic energy attack against an open web server as a case study, covering the attack vector discovery, exploitation, and detection avoidance.

3.1 Scenario Selection

A great variety of tactics can be used to mount energy targeted attacks against server systems. For example, if attackers obtain "root" or "administrator" privilege on a victim system, they can deliberately misconfigure drivers and/or firmware, e.g., overclock processor and memory, to operate the hardware components out-of-specs. Even with the privilege of a normal user, attackers can still easily increase the power consumption by running badly behaving programs such as a tight dead loop. However, the above-mentioned scenarios are not the focus of our study, because they are generally difficult to implement from remote (e.g., requiring privileged or physical access to the victim system).

We are interested in more commonly encountered scenarios, in which energy attacks can be launched without any special privileges. We assume that.

1. the victim server runs an open service, which accepts service requests from the Internet;
2. the attackers have no physical access to the victim server;
3. the attackers only have equivalent privileges of "anonymous users" on the victim server (for example, they cannot change system configurations or execute arbitrary code); and
4. there are no exploitable security vulnerabilities on the victim system to escalate the attackers' privileges.

In other words, the attackers communicate with the victim server using the same method as legitimate users, and the major variable they can manipulate is the server's workload, by crafting and submitting malicious service requests.

Thanks to the generic setting of attack environment, we believe that our scenarios are applicable to a wide range of servers, particularly, public web services such as news, blogs, forums, public data services including file and image sharing sites, and search engines.

3.2 Case Study: Wikipedia Mirror Server

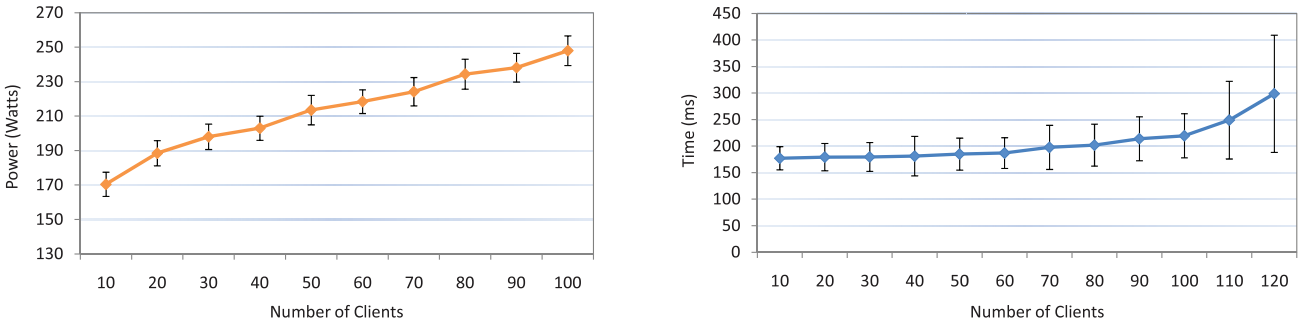
We perform a case study of designing an energy attack on an open web server. We use System B (i.e., the newer, energy-proportional server) as the victim, running a Wikipedia service with setup detailed in Section 4.1. We choose Wikipedia mirror as our attack target because it is a freely available, content-rich web service—a representative of real world production-use open web services.

3.2.1 Identifying an Attack Vector

The Wikipedia mirror is powered by MediaWiki, a large-scale content management system. The contents of all MediaWiki pages are stored in a marked up format different from standard HTML, and pages are dynamically generated when they are requested. Two levels of caching, "object cache" and memory cache, help to optimize the performance.

MediaWiki stores the dynamically generated HTML contents in an "object cache"—a database table. When a page is requested repeatedly, the HTML content is retrieved directly from the object cache without being repeatedly generated. A cached HTML page expires either after a period of inactivity or the associated page content has been modified. In addition to the object cache, the MySQL database speeds up operations by storing a portion of frequently queried table entries, as well as table search indices and query results in a memory, employing a modified LRU replacement algorithm.

We profile the power consumption and service latency characteristics of the two caching mechanisms on the target server, using the "Watts up? .Net" digital power meter. Figs. 2a and 2b show the average power usage and average response time for serving page requests from a single client in three different caching scenarios: pages being fully cached (in both memory and object cache), pages only in object cache, and pages not being cached. The lower bound of Y-axis in Fig. 2a is set to 130 watts, the system idleness power consumption. Thus, the columns in the figure represent the additional power consumption caused by the service requests.



(a) Power Consumption vs. Workload (b) Response Time vs. Workload

Fig. 3. Power consumption and response time profiles of victim server.

From this measurement, we can observe that compared with fully cached requests, requests with memory cache misses incur 3 percent power increase and 129 percent processing time increase, and requests with object cache misses incur 12.7 percent power increase and 840 percent processing time increase. Because energy is defined as the product of power and time, the effect of cache misses on energy consumption increase is multiplicative. The high energy cost rendered by cache misses forms an effective energy attack vector to our Wikipedia mirror server.

3.2.2 Exploiting the Attack Vector

Our next step is devising a method to exploit the discovered attack vector, that is, to generate requests that can cause cache misses, especially object cache misses. We examine previous studies in web browsing behaviors. According to Barford and Crovella [2], webpage accesses on a web server follow Zipf distribution, i.e., access frequency of a page correlates with its rank, and most accesses concentrate on a small number of pages while a large number of pages are rarely accessed. It is clear that the caching mechanisms in our web server work well in handling such an access pattern because they are designed to optimize for similar access patterns. However, this knowledge also hints at a practical cache attack scheme. To generate page requests with high probability of cache miss, we may access pages in patterns following a very different distribution from Zipf. For the ease of study and implementation, we choose a uniform random page access pattern to exploit our attack vector.

3.2.3 Detection Avoidance

The selected attack vector enables us to increase the victim's energy consumption without sending a large amount of requests. To avoid generating abnormal traffic patterns, we model the attacking request rate after "normal" web clients.

The study by Barford and Crovella [2] also shows that web browsing exhibits an "active-inactive" behavioral pattern. During the active period, a client submits requests in a bursty manner, which is attributed to the browser downloading multiple resources (images, scripts, etc.) linked to a document. During the inactive period, the client pauses sending requests, presumably because of the user reading the page content. The length of the inactive period follows Pareto distribution.

For our experiments, we simplify our model by "condensing" the active period into a single request, and only model the inactive period for request interarrival time. This is because all Wikipedia pages are text-oriented and structurally alike. The client behaviors in all the active periods would be very similar.

In addition to traffic shaping, we also need to adaptively adjust the injection of malicious requests based on the workload of the victim server. This is achieved by associating the victim server's workload with the response time. During the attack, we monitor the response time, with which we can infer the server's workload, and adjust the amount of malicious requests accordingly.

4 Attack Evaluation

In this section, we first describe the experimental setup. Then, we detail the preparation and measurements of the energy attack. And finally we assess the achievable damage.

4.1 Configuration and Setup

We set up a Wikipedia mirror server on System B using the classical LAMP combination (Linux, Apache, MySQL, and PHP). The database is imported from a Wikipedia dump containing 9,053,725 page entries. With a number of tests, we find that the server is capable of caching about 10,000 pages in memory. Therefore, we randomly pick 50,000 pages for use in our experiment.

We simulate client requests using a custom program running on a desktop computer. The client program simulates multiple clients each running in a separate thread. The "normal" clients are configured to access selected pages following Zipf distribution with $\alpha = 1$, and the request interarrival time follows Pareto distribution with $k = 1$ and $\alpha = 1.5$. The "malicious" clients are configured to access selected pages with the same request interarrival time distribution as the "normal" clients, but in a uniform random manner.

4.2 Workload—Response Time Profile

Before launching the attack, we first profile the victim server and establish the correlation between its workload and response time, as shown in Fig. 3b. Each data point is the average of 250 samples of service response time obtained under the corresponding workload. The error bar represents the standard deviation of response time. For light and moderate workloads (up to 50 clients), the server's response time increases quite slowly. When the workload increases beyond 60 percent, or 60 clients, the response time starts to

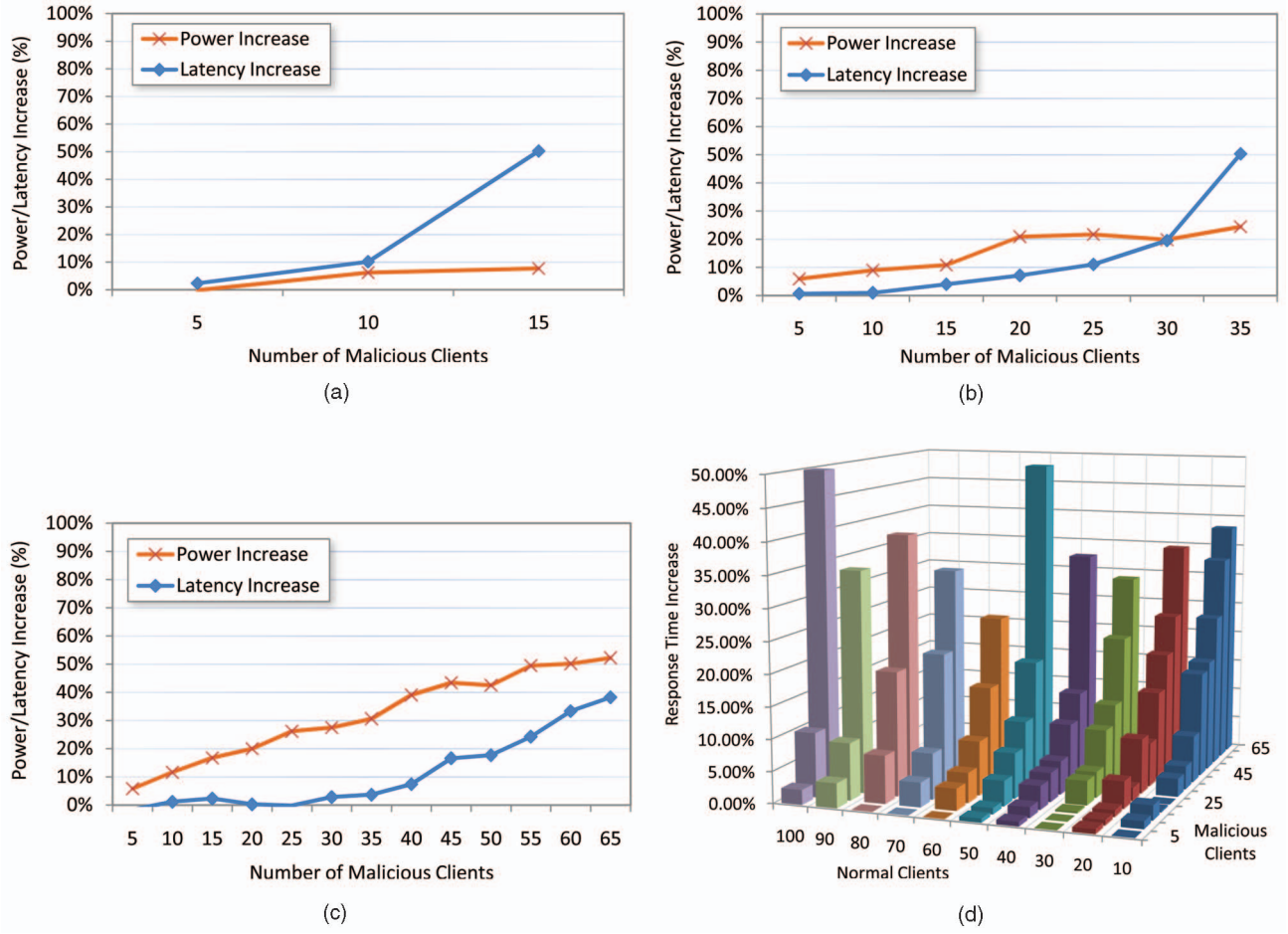


Fig. 4. Effects of attack under different Benign workload.

rise significantly. With workloads in which the number of active clients is beyond 100, the server starts to show symptoms of being overloaded—all clients experience intermittent short burst of request failures in the form of "HTTP 500" errors. Therefore, we determine that the server is capable of stably supporting up to 100 normal clients. Fig. 3a shows the correlation between stable workload and system power consumption, from which we can see that the server system power consumption is indeed proportional to its workload.

4.3 Attack Measurements

We use server-side power consumption and client-side perceived response latency to measure the effects of the energy attack. We conduct the experiments using different server workloads, which range from 10 to 100 normal clients with the increment of 10 clients. For each workload, we inject energy attack traffic by adding a number of malicious clients. Due to the large volume of data, we only present the results corresponding to 100, 50, and 10 normal clients and depict them in Figs. 4a, 4b, and 4c, respectively. These figures show the increases in power consumption and response latency caused by the introduction of malicious workloads.

At 100 percent of the full load, as shown in Fig. 4a, the response latency of the victim server is very sensitive to the addition of malicious workloads. The malicious workload of 10 malicious clients increases the response latency by 7.6 percent, and the workload of 15 malicious clients increases the response latency by 50.2 percent. The power consumption, however, does not increase with the response latency, as the server is already fully loaded.

At 50 percent of the full load, as shown in Fig. 4b, with 20 malicious clients, the attack results in 20.9 percent of extra power being consumed while only incurs 7.1 percent increase in response latency. However, with 30 or more malicious clients, the response latency increase surpasses the power consumption increase.

At 10 percent of the full load, as shown in Fig. 4c, the energy increase caused by the attack becomes very significant. With 40 malicious clients, the victim server's power consumption increases by 39.0 percent, while the service response latency only increases by 7.4 percent.

4.4 Damage Assessment

Our measurement results show that, at any stable workload, energy attacks will cause increased power consumption on the victim server. The more malicious clients, the larger the power increase. However, a larger number of malicious clients also results tangible performance degradation. Fig. 4d presents the collective results of service response time increases for all ten different workloads with varying numbers of malicious clients. Note that samples with response time increment larger than 50 percent are omitted due to their unimportance.

To guarantee the success of an energy attack, low attack profile takes precedence over the power consumption increment. Therefore, the number of malicious clients

TABLE 2
Percentage of Power Increases due to Attack

Utilization	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Power Increase	39.0%	42.3%	36.3%	31.6%	21.7%	14.8%	11.6%	9.0%	11.3%	6.2%

needs to be limited to avoid significant response time impact. We refer to the workload—response time profile for a reasonable threshold. The standard deviation of response time at stable workloads varies between 12.3 and 21.1 percent of the measured values. We thus use the smallest percentage, 12.3 percent, as the upper limit of response time increases. With the chosen constraint, we determine the maximum power consumption achievable by the attacks for each workload, and present them in Table 2. We observe that, the power increase effect of the energy attack is inversely correlated with the benign workload of the server—an idle server suffers significant extra power consumption, while a very busy server only incurs a small power consumption increase.

To assess the nominal damage of this energy attack to a server, we refer to the study of typical server workloads. Barroso and Hölzle [4] observe that most servers have average utilization between 10 and 50 percent. Correspondingly, under such utilization, our energy attack can result in 21.7-42.3 percent power consumption increase.

5 Defending against Energy Attacks

The high potential damage of the energy attacks calls for an effective defense. In this section, we first discuss the difficulties in defending against energy attacks. Then, we present our solution to meet the challenges, and finally we validate the effectiveness of our defense scheme.

5.1 Defense Challenges

To defend against energy attacks, it is necessary to differentiate malicious users from benign users, by the amount of energy consumed in serving their requests. Unfortunately, although the power consumption of the whole system can be measured in a coarse time granularity, today's servers are unable to provide fine-grained power consumption measurement due to the lack of hardware support. As a result, currently it is not possible to measure and account for the actual power consumption of servicing each individual request. And consequently, it is a very challenging task to devise an effective and generalized protection mechanism against energy attacks.

One may be tempted to suggest detecting an energy attack by other metrics in place of fine-grained power instrumentation. For example, the energy attack used in our case study can be uncovered by detecting abnormal page visit patterns, instead of referring to power consumption measurements. However, this naïve solution suffers in terms of soundness and completeness. First, the cause-and-effect relationships are not definitive, and thus detecting an energy attack by other metrics may lead to high false positives. For instance, our case study exploits an abnormal page visit pattern. But not all page visit patterns that deviate from the norm result in energy attacks. Second, energy attacks could exploit a great variety of alternative attack vectors, and render the monitoring system ineffective. Unlike buffer overflow or code injection vulnerabilities, the energy security issue is rooted deep in the server system's design, and it can manifest itself as very different, unrelated attack vectors. We discuss two alternative attack vectors in Section 6.1, and the exploitation of each attack vector requires a separate metric to discover.

Compared with the aforementioned "side-metric" monitoring strategy, a more holistic approach is to build the defense system based on fine-grained power consumption information, and then derive the needed information by measuring related metrics. Neugebauer and McAuley [26] suggest using performance counter data such as CPU cycles, disk operations, and screen pixels to approximate power consumption for laptops and mobile devices. Buennemeyer et al. [6] present a battery-sensing intrusion protection system for mobile computers, which correlates device power consumption with Wi-Fi and Bluetooth communication activities. Kim et al. [22] propose a power-aware malware detection framework by collecting application power consumption signatures. While this approach achieves good generalization, it suffers low accuracy on server systems. This is because mobile devices are operated by individuals, and they run few applications concurrently. In contrast, server systems are designed to process a large number of requests from multiple users in parallel. As a result, performance counter readings of independent request-serving processes (especially at fine granularity) can be heavily coupled and inaccurate for power approximation. For example, processors in a server share underlying hardware, such as the memory bus and PCI devices. And unrelated processes competing for shared resources can lead to heavy interferences of each other's cycle count readings. For another example, modern hard drives can intelligently reorder the sequence of operations to improve efficiency. However, this optimization can cause the operation latency disproportional to the complexity of a data request.

5.2 Energy-Aware Programming

The lack of hardware support makes fine-grained power measurement on today's server systems unachievable. In addition, the parallel processing nature of request servicing renders low-level counter-based power approximation inaccurate. To work around these limitations and enable the design of an effective and generalized defense system, we take an application-oriented approach and propose energy-aware programming.

The key idea of energy-aware programming is to capture the power consumption of individual request servicing in the form of application code execution, and enable an application to differentiate power consumption of service requests. Energy-aware programming infers

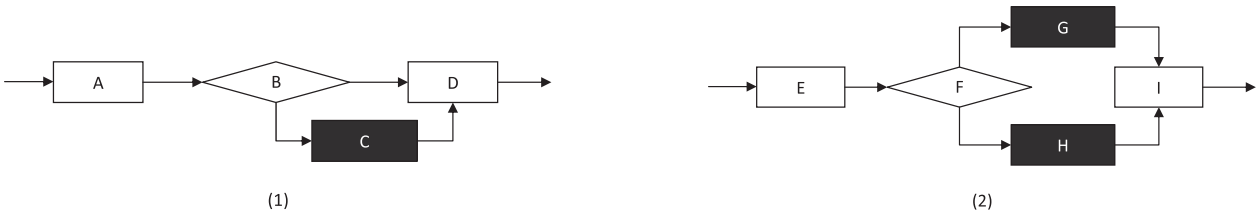


Fig. 5. Example component flow chart.

power consumption information by leveraging high-level application context. Thus, this technique is much less prone to interferences than those low-level counter-based power approximation techniques.

5.2.1 Power Consumption Characterization

In order to create an energy-aware application, application developers need to characterize the power consumption of the components in their applications, and embed such information into the program code. This can be accomplished in three steps.

The first step is the analysis and collection of conditionally invoked components. As shown by the two example flow charts in Fig. 5, components C, G, and H are conditionally invoked while all others are mandatory. The rationale behind this design is that the mandatory executed code contributes to the *baseline* power consumption for all request services, but the conditionally invoked code is responsible for the *dynamic* power consumption, which can increase dramatically for servicing power-extensive requests.

The second step is to characterize the power consumptions of component collection $\{C_1, C_2, \ldots, C_n\}$. This can be done using basic profiling techniques. Given a fixed time interval t and a specific number of invocations r, each component C_i is invoked r times in t seconds, and the average system power consumption $Psys_i$ is measured. The effective power consumption P_i is then derived by contrasting $Psys_i$ with the system idle power consumption P_{idle}. The power consumption readings do not have to be very precise, because 1) in absence of fine-grained hardware support, it can be difficult to obtain accurate power consumption readings; and 2) the goal of this profiling is to differentiate components by their power consumption, and then use this information to infer the system dynamic power state and the nature of future workload.

The final step is to annotate the power information in the application. This is fulfilled by embedding the component power consumption table and inserting *power counters* into the program. For each component C_i, a counter I_i is assigned, and each invocation of the component results in an increment of its associated counter.

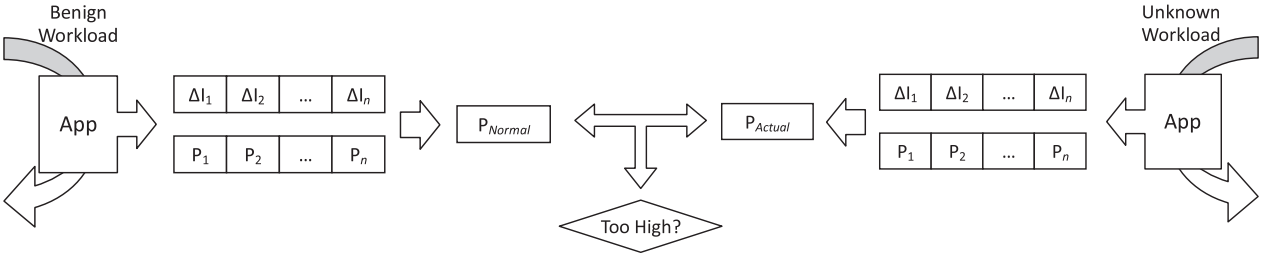
5.2.2 Power State Inference

With embedded power information in an application, the *dynamic* power state (`Power` for short) can be inferred by computing $P = \sum(P_i \cdot \Delta I_i)$, referring to the embedded component power consumption table and the increments of *power counters*. This calculation applies to the entire system as well as individual request servicing. In other words, the `Power` of the whole system at any given interval can be calculated by collecting the *power counter* increments during that interval; and the `Power` for servicing a specific request can be calculated by using the *power counter* increments caused by servicing this request. Therefore, an energy-aware application can self-monitor its power states at coarse-grained and fine-grained levels, and thereby is capable of detecting energy attacks, identifying attackers, and reacting accordingly.

5.3 Defense System Design

With the help of energy-aware programming, we design a simple and effective defense system to shield applications from energy attacks. The system is composed of three components, energy attack detection, power history maintenance, and defensive reaction.

The first component is responsible for detecting energy attacks on the server system. Before an energy-aware application is deployed, we first subject it to benign workloads, and record the system's normal `Power`, P_{Normal}, as shown in Fig. 6a. After the application has been deployed, the attack detection component monitors the `Power` of the entire system, and compares it with P_{Normal}. As shown in Fig. 6b, when the system power consumption surges significantly above the normal value, it asserts that the system is under an energy attack.



(a) Measure Normal Power Consumption (b) System-wide Power Attack Detection

Fig. 6. System power measurement and attack detection.

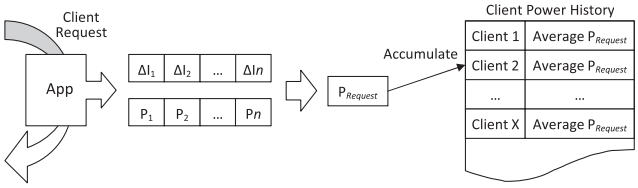


Fig. 7. Maintaining client power history for defensive throttling.

The second component is used for maintaining client power history records. Illustrated in Fig. 7, for each communicating client, its average power consumption for request servicing is maintained. During an energy attack, these records are used by the defensive reaction component as a reference to classify malicious clients. To scale with the client population growth, if necessary, we can maintain the average power history record per subnet or per domain by aggregating a group of clients into a single cluster.

The third component is designed for providing defensive reactions when an energy attack is detected. It sorts the power history records, and identifies the clients on top of the sorted list as malicious, because they represent the major sources of increased power consumption. It then applies defensive operations to these clients to reduce their energy impacts. For example, throttling down or blocking their request servicing for a period of time until the system is no longer under the energy attack. The choice of defensive reaction is flexible, and can be fully customized depending on the tolerance of false positives.

Thanks to energy-aware programming, our defense system can uncover the stealthiness of malicious energy attack clients, and thus prevent potential evasion attempts. An attacker may utilize a large number of compromised machines (such as a botnet) and make each client very low profile. However, the attacking clients can still be identified and reacted upon by our defense system. This is because the service requests from each malicious client, although low in intensity, still consist of a high percentage of power-consuming service requests.

Although determined attackers might evade our defense system by lowering the concentration of malicious requests, and thus making a malicious client's power consumption comparable to that of a benign client. However, doing so would also significantly reduce the effectiveness of the energy attack, and require the attacker to exploit a much larger number of compromised clients and to generate much higher traffic volume to achieve the same effect. Such a practice would degrade an energy attack to a regular Distributed DoS (DDoS) attack, which can be defended by various previously proposed work [21], [31], [36]. The discussion of defending against DDoS attacks is beyond the scope of this paper.

5.4 Defense Experiment

To validate the effectiveness of our defense scheme, we have implemented a prototype defense system in our Wikipedia mirror server. In the following, we first briefly describe the implementation and then present the experimental results.

According to our profile measurements in Section 2, an object cache miss in page request servicing incurs high power consumption. We thus analyze the MediaWiki page request handling routine, and present its abstract flow chart in Fig. 8. The main difference between object cache hit and miss lies in the invocation of the "Parser" component, which performs a processor intensive operation that dynamically generates HTML content from the Wiki style mark up text. Consequently, we can build an effective defense against energy attacks by protecting this single component.

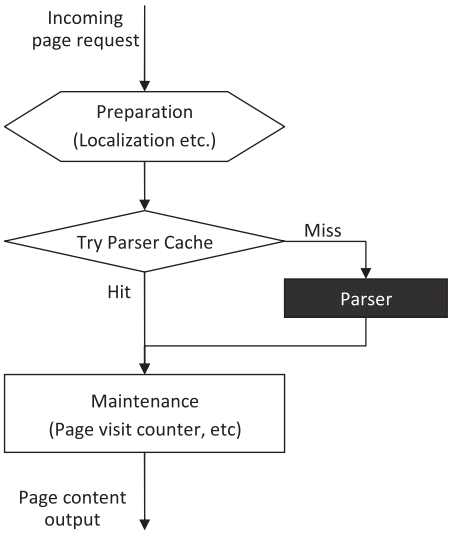


Fig. 8. MediaWiki flow chart.

We augment MediaWiki with "energy-awareness" by counting the parser invocations. Since there is only one power-extensive component, we simply omit power profiling, and assign 1.0 as this component's symbolic `Power`. That is, if the parser is invoked while serving a page, the application's *dynamic* power consumption is 1.0; otherwise, the *dynamic* power consumption is 0.0 . Thus, the `Power` we are interested in can be expressed as the parser invocation ratio.

The storage and computation overhead of the defense system is minor. For each client, a 12-byte power history record is used to maintain the client's identity, number of page visits, and *dynamic* power consumption. For each request servicing, a hash table look-up is performed to retrieve client's power history record, and then several arithmetic operations are performed to update the record. Overall, the overhead is negligible compared with the storage and processing time required to serve a Wikipedia page.

We first test our defense system's ability to differentiate benign and malicious clients, as well as collecting the system's normal *dynamic* power consumptions for defense purposes. When we subject the server to benign requests, we observe that the system *dynamic* power stays around 0.04-0.05. However, when we inflict malicious requests on the server, the system *dynamic* is increased to 0.6-0.8. We heuristically set the system normal *dynamic* power to 0.3, and set the protection threshold to be 30 percent parser invocations with 128 accesses. Therefore, when the victim server's system wide *dynamic* power exceeds 0.3, users will be throttled down if they have over 128 recorded accesses, and have triggered the page parser invocation over 30 percent of the time.

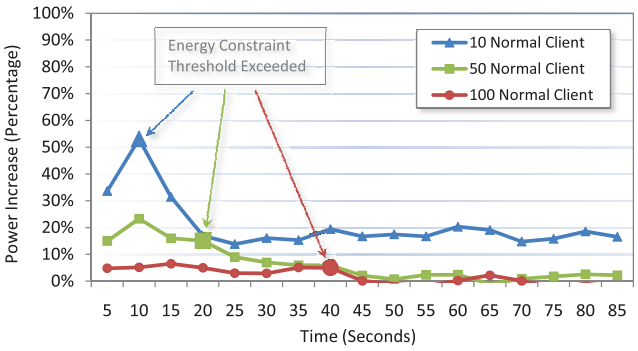


Fig. 9. Defense of attack at different workloads.

We then configure and activate our defense system online. We measure the actual power consumption increases of the victim server under the same energy attack, and present the results in Fig. 9. We use the same server workloads in Section 4. For 10 normal clients (i.e., low workload), the increase of server power consumption reaches about 50 percent at the beginning of attack. However, the attacker is unable to sustain the high power consumption increase. The parser invocation ratio quickly exceeds the threshold of our defense system, and the attacker's service requests are discarded afterwards. Correspondingly, the power consumption increase of the server is reduced to 19 percent. We can observe similar effects for 50 and 100 normal clients—our defense lowers the power consumption increments from 21 to 4 percent, and from 6 percent to near zero, respectively. Meanwhile, page accesses from normal clients at all workloads are served without interruption.

6 Discussion

In this section, we first describe two other possible energy attack vectors, then discuss the applicability limitation of energy attacks, and finally discuss the limitation of energy-aware programming and the potentials of hardware assisted defenses against energy attacks.

6.1 Attack Variations

In addition to using cache miss as an attack vector, energy attacks can also be launched by exploiting other energy related vulnerabilities. For example, a file depositing server running an unmodified Linux kernel and allows users to control the names for stored files (such as a public FTP server) is vulnerable to energy attacks. The attacker can exploit a well-known *nix kernel file name resolution vulnerability and launch a low-rate algorithmic complexity attack [7], [10] to stealthily increase processor utilization. Because a file depositing service is storage and network bandwidth bound, a well-controlled energy attack can avoid generating any throughput anomalies.

Besides the processors, other components with large dynamic power range can also be exploited by energy attacks. For example, hard drives normally consume 12 to 16 watts during operation, but their power consumption can be reduced to under 1 watt by spin-down the platters during long period of idleness. As a result, an energy attack on hard drives can be mounted by performing sleep deprivation attack [23], [28] to prevent expected spin-down. Although the energy cost of a single attacked hard drive seems to be insignificant, the damage can accumulate to a significant amount when the energy attack targets at a decent sized storage server with 10 to 20 installed hard drives.

6.2 Attack Applicability

We have thoroughly investigated the proposed energy attack against a stand-alone server system. We use the case of single stand-alone server as the first step to study energy attack, because it is relatively easy to perform a clear analysis and repeatable evaluations. However, the attack vectors on a stand-alone server are not applicable to other hosting configurations, such as clustered servers and load balanced server farm. For example, our proposed energy attack on our Wikipedia mirror server is not effective on the actual Wikipedia website, which employs load balanced server clusters and heavy proxy caching techniques. In order to launch energy attack against a service configured in multiserver setup, one needs to discover and exploit new attack vectors.

Nevertheless, we believe energy attacks also pose serious threats to large scaled systems. For example, in a cloud hosting environment [14], competing cloud vendors may use energy attack as a powerful weapon to increase the operation cost of their opponents, and make the attackers' service rates more attractive. To extend the scope of this work, we plan to study and profile the interactions of workload and power consumption of server clusters, discover viable attack vectors, as well as devise defending techniques.

6.3 Limitation of Defense

Even as an effective workaround of the missing hardware support, we acknowledge that energy-aware programming places the nontrivial duty of power profiling and energy accounting onto the application developers. For more scalable and accurate solutions, we advocate enabling fine-grained power measurement at the hardware and operating system level, making energy consumption information as accessible as the performance data. For example, the processor can include an "energy counter" similar to performance counters, and account the amount of energy consumed by a particular thread at given time period, based on the amount and variety of circuitry being activated.

7 Related Work

As energy cost of server systems takes a significant proportion of IT expenditures, research on power management techniques for server systems has been very active in recent years. A survey of power and energy research for server systems can be found in [5]. Elnozahy et al. [13] present two power management mechanisms—dynamic voltage scaling (DVS) and request batching—to reduce energy consumption in web servers. Horvath et al. [19] explore the benefits of using DVS in multistage service pipelines for power management in server farms. Felter et al. [15] study power shifting, which reduces peak power with minimal performance impact by dynamically reallocating power to performance critical components. Meisner et al. [24] design a system called

Powernap, which can reduce server idle power by rapidly transitioning the entire system between an active state and a near-zero-power idle state.

In [4], Barroso and Hölzle present the concept of Energy-Proportional Computing. They call for improvements in the energy usage profile of every system components, particularly the memory and disk subsystems to achieve energy proportionality. Barroso and Hölzle also point out that server systems may not benefit as much as the mobile systems from the energy-efficiency schemes targeting mobile devices, due to the distinct behavior of server workloads. To make the entire server system energy efficient, energy proportionality must be included in the design objectives for each component.

Energy management in server clusters has also been extensively studied. Chase et al. [9] design a resource management system for called Muse with a primary focus on energy for large server clusters. Muse promotes energy efficiency of server clusters by balancing the cost of resources against the achieved benefit. Pinheiro et al. [27] propose a load concentration technique that can dynamically distribute the load and set some hardware resources in low-power modes to conserve energy. Elnozahy et al. [12] evaluate different combinations of cluster reconfiguration and dynamic voltage scaling. Rajamani and Lefurgy [30] investigate the key factors in the system-workload context that affect energy saving policies in server clusters. Heath et al. [18] study the energy conservation in heterogeneous server clusters using a model-based cooperative web server. Fan et al. [14] present the aggregate power usage characteristics of several large-scale workloads from a data center over a period of six months, and they find that the opportunities for energy savings at the cluster level are greater than at the rack level.

Different from the research on power management in server systems that mainly focus on energy conservation, the security issue of power and energy has gained much attention in mobile computing community, because the power of battery is a critical and scarce resource for mobile computing devices. In [11], Dagon et al. categorize a number of security problems caused by mobile malware and point out that battery exhaustion, a type of DoS attacks, is a serious threat to mobile computing. Three types of battery depletion attacks are presented in [23]. Racic et al. [29] demonstrate that the attack that on mobile phones' battery can be stealthily launched by exploiting the vulnerability of cellular service Multimedia Messaging Service (MMS) and that the attack can drain the power of batteries up to 22 times faster.

A number of research efforts have been spent in detecting and preventing attacks on battery power of mobile devices. Martin et al. [23] propose a power-secure architecture, which employs multilevel authentication and energy signatures, to counter power attacks. Buennemeyer et al. [6] present a battery-sensing intrusion protection system for mobile computers that correlates device power consumption with Wi-Fi and bluetooth communication activities. Kim et al. [22] propose a power-aware malware detection framework that can detect previously unknown energy-depletion attacks by collecting power consumption information of applications and comparing their power signatures with the signatures of normal applications.

8 Conclusion

Server systems have become more power efficient and energy proportional as power management technologies advance. However, the security aspect of power management has not yet been studied. In this paper, we investigated the potential vulnerabilities in server power management. First, we exposed the threat of energy attacks by measuring the power consumption of real server systems. Then, we designed and evaluated an energy abusing attack on server systems. In particular, we validated the threat of energy attacks on an open web server running Wikipedia mirror service. By profiling power consumption of the target server under different operation conditions, we realized a viable energy attack vector. We conducted a series of experiments, in which energy attacks with varying attack intensities were carefully mounted to avoid incurring tangible degradation of server performance. Our experimental results show that the proposed energy attack can incur significant increase of power consumption on the victim server. Finally, we presented an application-oriented defense approach to work around the current limitations of the hardware. Our evaluation shows that this software-based defense scheme is effective in protecting victim servers against energy attacks.

Acknowledgments

The authors would like to thank the anonymous reviewers for their detailed and insightful comments, which have helped to greatly improve the quality of the paper. This work was partially supported by US National Science Foundation (NSF) grant 0901537 and ONR grant N00014-09-1-0746.

References

[1] "Advanced Configuration and Power Interface," http://www.acpi.info, 2009.

[2] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," *Proc. ACM SIGMETRICS Joint Int'l Conf. Measurement and Modeling of Computer Systems,* pp. 151-160, 1998.

[3] L.A. Barroso, "The Price of Performance," *ACM Queue,* vol. 3, no. 7, pp. 48-53, Sept. 2005.

[4] L.A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *Computer,* vol. 40, no. 12, pp. 33-37, Dec. 2007.

[5] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *Computer,* vol. 37, no. 11, pp. 68-74, Nov. 2004.

[6] T.K. Buennemeyer, M. Gora, R.C. Marchany, and J.G. Tront, "Battery Exhaustion Attack Detection with Small Handheld Mobile Computers," *Proc. IEEE Int'l Conf. Portable Information Devices (PORTABLE),* 2007.

[7] X. Cai, Y. Gui, and R. Johnson, "Exploiting Unix File-System Races via Algorithmic Complexity Attacks," *Proc. IEEE 30th Symp. Security and Privacy,* May 2009.

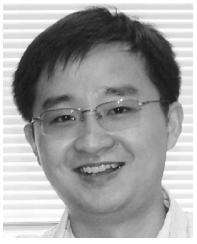
[8] E.V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," *Proc. 17th Ann. Int'l Conf. Supercomputing (ICS),* pp. 86-97, 2003.

[9] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle, "Managing Energy and Server Resources in Hosting Centers," *Proc. 18th ACM Symp. Operating Systems Principles (SOSP),* pp. 103-116, 2001.

[10] S.A. Crosby and D.S. Wallach, "Denial of Service via Algorithmic Complexity Attacks," *Proc. 12th Conf. USENIX Security Symp.,* 2003.

[11] D. Dagon, T. Martin, and T. Starner, "Mobile Phones as Computing Devices: The Viruses Are Coming!" *IEEE Pervasive Computing,* vol. 3, no. 4, pp. 11-15, Oct.-Dec. 2004.

[12] M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," *Proc. Second Workshop Power-Aware Computing Systems,* pp. 179-196, 2002.
[13] M. Elnozahy, M. Kistler, and R. Rajamony, "Energy Conservation Policies for Web Servers," *Proc. Fourth Conf. USENIX Symp. Internet Technologies and Systems (USITS),* 2003.
[14] X. Fan, W.-D. Weber, and L.A. Barroso, "Power Provisioning for a Warehouse-Sized Computer," *Proc. 34th Ann. Int'l Symp. Computer Architecture (ISCA),* pp. 13-23, 2007.
[15] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems," *Proc. 19th Ann. Int'l Conf. Supercomputing (ICS),* pp. 293-302, 2005.
[16] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," *Proc. 30th Ann. Int'l Symp. Computer Architecture (ISCA),* pp. 169-182, 2003.
[17] J. Hamilton, "Where Does the Power Go and What to Do About It?" *Proc. USENIX Workshop Power Aware Computing and Systems (HotPower),* 2008.
[18] T. Heath, B. Diniz, E.V. Carrera, W. Meira Jr., and R. Bianchini, "Energy Conservation in Heterogeneous Server Clusters," *Proc. 10th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP),* pp. 186-195, 2005.
[19] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic Voltage Scaling in Multitier Web Servers with End-to-End Delay Control," *IEEE Trans. Computers,* vol. 56, no. 4, pp. 444-458, Apr. 2007.
[20] *Intel 6400/6402 Advanced Memory Buffer: Thermal/Mechanical Design Guide,* Intel, Dec. 2006.
[21] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds," *Proc. Second USENIX Symp. Networked Systems Design and Implementation (NSDI),* May 2005.
[22] H. Kim, J. Smith, and K.G. Shin, "Detecting Energy-Greedy Anomalies and Mobile Malware Variants," *Proc. Sixth Int'l Conf. Mobile Systems, Applications, and Services (MobiSys),* pp. 239-252, June 2008.
[23] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-Service Attacks on Battery-Powered Mobile Computers," *Proc. IEEE Second Int'l Conf. Pervasive Computing and Comm. (PerCom),* 2004.
[24] D. Meisner, B.T. Gold, and T.F. Wenisch, "PowerNap: Eliminating Server Idle Power," *Proc. 14th ACM Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS),* pp. 205-216, Mar. 2009.
[25] R. Nathuji and K. Schwan, "VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems," *Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP),* pp. 265-278, 2007.
[26] R. Neugebauer and D. McAuley, "Energy Is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS," *Proc. Eighth Workshop Hot Topics in Operating Systems (HOTOS),* 2001.
[27] E. Pinheiro, R. Bianchini, E.V. Carrera, and T. Heath, *Dynamic Cluster Reconfiguration for Power and Performance,* pp. 75-93. Kluwer Academic Publishers, 2003.
[28] M. Pirretti, S. Zhu, V. Narayanan, P. Mcdaniel, and M. Kandemir, "The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense," *Proc. Innovations and Commercial Applications of Distributed Sensor Networks Symp. (ICA DSN),* 2005.
[29] B.R. Racic, D. Ma, and H. Chen, "Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone's Battery," *Proc. Second Int'l Conf. Security and Privacy in Comm. Networks (SecureComm),* pp. 1-10, Sept. 2006.
[30] K. Rajamani and C. Lefurgy, "On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software (ISPASS),* pp. 111-122, 2003.
[31] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "DDoS-Resilient Scheduling to Counter Application Layer Attacks Under Imperfect Detection," *Proc. IEEE INFOCOM,* Apr. 2006.
[32] Seagate, "Barracuda ES.2 Data Sheet," http://www.seagate.com/docs/pdf/datasheet/disc/ds_barracuda_es_2.pdf, 2012.
[33] Seagate, "Cheetah 15K.6 Data Sheet," http://www.seagate.com/docs/pdf/datasheet/disc/ds_cheetah_15k_6.pdf, 2012.
[34] US Environmental Protection Agency, "Report to Congress on Server and Data Center Energy Efficiency," 2007.
[35] US Environmental Protection Agency, "The ENERGY STAR Version 5.0 Specification for Computers," 2008.
[36] H. Wang, C. Jin, and K.G. Shin, "Defense Against Spoofed IP Traffic Using Hop-Count Filtering," *IEEE/ACM Trans. Networking,* vol. 15, no. 1, pp. 40-53, Feb. 2007.
[37] Watts Up?, "Watts Up? .Net Digital Power Meter," https://www.wattsupmeters.com/secure/products.php?pn=0, 2009.



Zhenyu Wu received his PhD degree in Computer Science from the College of William and Mary, Williamsburg, in 2012. He is a research staff member at NEC Laboratories America Inc., Princeton, New Jersey. His research focuses on enterprise system security and mobile application security. His research interests also lie in general system and network security, including but not limited to malware analysis, packet filters, and Internet chat and online game security. He is a member of the IEEE.

Mengjun Xie (S'08-M'10) received the PhD degree in computer science from the College of William and Mary, Williamsburg, in 2009. He is an assistant professor of computer science at the University of Arkansas at Little Rock, Little Rock. His research interests include network security, information security, network systems, and operating systems. He is a member of the IEEE.

Haining Wang received the PhD degree in computer science and engineering from the University of Michigan at Ann Arbor in 2003. He is an associate professor of computer science at the College of William and Mary, Williamsburg, Virginia. His research interests lie in the areas of security, networking systems, and distributed computing. He is a senior member of the IEEE.