The Security and Privacy Implications of Energy-Proportional Computing

A Dissertation Presented

by

SHANE S. CLARK

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

June 10, 2013

School of Computer Science

The Security and Privacy Implications of Energy-Proportional Computing

A Dissertation Presented

by

SHANE S. CLARK

Approved as to style and content by:

Kevin Fu, Chair

Wayne P. Burleson, Member

Deepak Ganesan, Member

Erik Learned-Miller, Member

Wenyuan Xu, Member

Lori A. Clarke, Chair
School of Computer Science

For Shahan.

ACKNOWLEDGEMENTS

First, I owe my sincere gratitude to my advisor, Kevin Fu, for guidance, ideas, funding, and unflagging support of new research ideas. His tutelage dates all the way back to my first research experience as an undergraduate.

I thank my coauthors for all of the hard work, great ideas, and valuable discussions over the years.

I also thank my peers in the SPQR group at UMass (and now at Michigan as well): Ben Ransford, Andrés Molina-Markham, Mastooreh Salajegheh, Amir Rahmati, Denis Foo Kune, Shane Guineau, Katarzyna Olejnik, and Hong Zhang. I also thank Jacob Sorber, the UMass ECE contingent under the guidance of Wayne Burleson, and the members of Lab lab.

I thank Bill Stasny and Hans Jensen for building and patiently helping me understand much of the outlandish hardware I have devised for this work.

I would also like to thank Laurie Downey, Leeanne Leclerc, and the rest of the CS staff who kept me on track.

Thanks to the Institute for Information Infrastructure Protection for the first-year scholarship and to the National Science Foundation for the Graduate Research Fellowship that allowed me to focus on research instead of how many packs of ramen noodles it is safe to eat each week.

I thank my family for always setting high standards and having little tolerance for excuses. I thank all of those who have helped me maintain my sanity outside of the lab, particularly Erik, Bill, Travis, Jeremy, Alyssa, Elias, Jeff, and Timur. Finally, I thank Shahan for all of the support that made it possible for me to survive grad school.

Thanks to all those who contributed to the publications included in this thesis as early readers, reviewers, shepherds, and thanks to the entities that funded the research. Portions of this thesis appeared in the following publications:

- "Potentia est Scientia: Security and Privacy Implications of Energy-Proportional Computing" by Shane S. Clark, Benjamin Ransford, and Kevin Fu. In Proceedings of the 7th USENIX Workshop on Hot Topics in Security (HotSec), Bellevue, WA, August 2012.
- "WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices" by Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. In Proceedings of the USENIX Workshop on Health Information Technologies (HealthTech), Washington, D.C., August 2013.
- "Current Events: Identifying Webpages by Tapping the Electrical Outlet" by Shane S. Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. In Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS), London, U.K., September 2013.

ABSTRACT

The Security and Privacy Implications of Energy-Proportional Computing

June 10, 2013

SHANE S. CLARK

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Kevin Fu

The parallel trends of greater energy-efficiency and more aggressive power management are yielding computers that inch closer to energy-proportional computing with every generation. Energy-proportional computing, in which power consumption scales closely with workload, has unintended side effects for security and privacy. Saving energy is an unqualified boon for computer operators, but it is becoming easier to identify computing activities by observing power consumption because an energy-proportional computer reveals more about its workload.

This thesis demonstrates the potential for system-level power analysis—the inference of a computers internal states based on power observation at the "plug." It also examines which hardware components and software workloads have the greatest impact on information leakage. This thesis identifies the potential for privacy violations by demonstrating that a malicious party could identify which webpage from

a given corpus a user is viewing with greater than 99% accuracy. It also identifies constructive applications for power analysis, evaluating its use as an anomaly detection mechanism for embedded devices with greater than 94% accuracy for each device tested. Finally, this thesis includes modeling work that correlates AC and DC power consumption to pinpoint which components contribute most to information leakage and analyzes software workloads to identify which classes of work lead to the most information leakage.

Understanding the security and privacy risks and opportunities that come with energy-proportional computing will allow future systems to either apply system-level power analysis fruitfully or thwart its malicious application.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

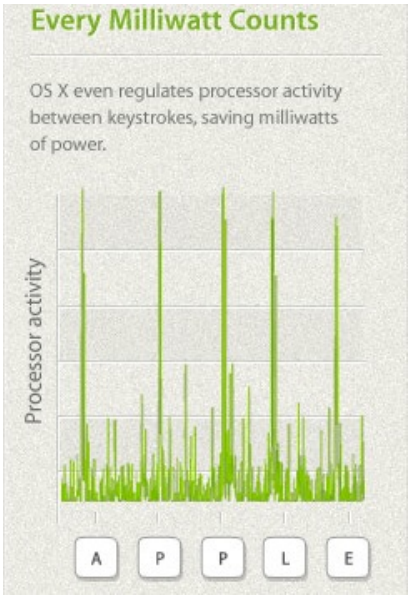
INTRODUCTION

As computing power and efficiency improve in tandem, device power consumption is increasingly proportional to workload. Saving energy is an unqualified boon for computer operators, but this trend has produced an unintentional side effect: a modern computer's power consumption, even observed at the granularity of a complete system, reveals detailed information about its workload. Intuitively, reducing idle power consumption lowers the effective noise floor, making activity patterns "stick out" more by raising the signal-to-noise ratio. The information leaked by these activity patterns can be leveraged to mount attacks against user or system privacy, or to ensure that device behavior conforms to expectations.

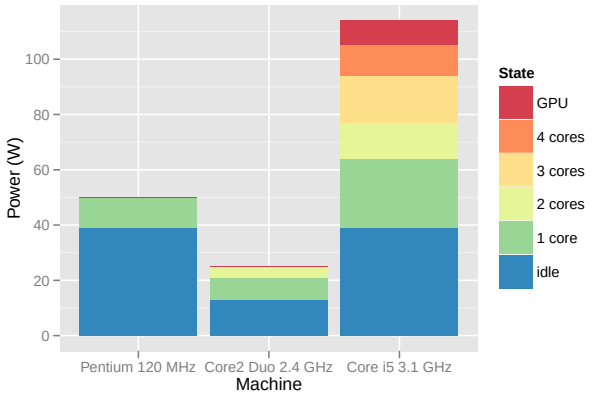
1.1 Background and Motivation

The parallel trends of greater energy-efficiency and more aggressive power management are yielding computers that inch closer to energy-proportional computing with every generation. An energy-proportional computer is one for which power consumption scales closely with workload. Unlike simple electronics that may have only "on" and "off" states, energy-proportional computers have a wide power range [8], with consumption closely mirroring workload fluctuations. Figure 1.1b illustrates the increasing energy proportionality of 3 commodity computers. The newer computers show obvious power consumption scaling as each CPU core is put under load.

Dynamic voltage and frequency scaling (DVFS), clock gating, turbo modes, and dark silicon [21] are all recent examples of hardware optimizations made in the service



(a) An Apple advertisement from 2009 [6] touts energy-efficiency gains that also happen to reveal keystrokes in power traces.



(b) An illustration of increasing energy proportionality for 3 computers. The oldest computer's power consumption changes very little with resource consumption, but the newest computer's power consumption more than doubles in response to workload changes.

Figure 1.1

of increasing both energy efficiency and performance for different workloads. Modern CPUs and GPUs supporting both clock gating and turbo modes, for example, can power down one or more processor cores and increase the clock speed on others to maximize single-threaded performance without violating thermal design power (TDP) limits.

Both Intel and AMD have also begun to integrate CPUs and GPUs on the same physical chip in the interest of energy savings. Tighter physical coupling allows designers to use fewer transistors by eliminating redundancies and simplifying data sharing. AMD even markets their tightly integrated architectures as Application Processing Units (APUs) [79] rather than CPUs, touting their promise as platforms for heterogeneous computing frameworks such as OpenCL [41].

While techniques such as DVFS and clock gating are relatively new, the trend toward greater energy efficiency extends far into the past. Koomey et al. point

out that energy efficiency (computations per kilowatt-hour) has doubled every 1.57 years from 1946 to 2009, with much room for improvement remaining—seven orders of magnitude, extrapolating from an estimate by Feynman—until designs hit theoretical limits [45].

Energy efficiency is in some ways a necessity for CPUs and GPUs because of TDP constraints, but the designers of other computer hardware also have incentives to improve energy efficiency through government campaigns such as ENERGY STAR [73] or the desire to advertise eco-friendliness as a feature. Western Digital, for example, has a special line of eco-friendly hard drives branded as "Caviar Green" [81].

The growing consumer dependence on battery-constrained mobile devices has also driven the adoption of aggressive power management techniques in software. New operating systems leverage the low power modes offered by modern hardware to cut power consumption during periods of low activity. Windows 8, for example, even mandates the suspension of most "Windows Store apps" that do not have focus to save power [23]. Similarly, the webpage for the Apple Mac Mini [6] claims that the operating system

> *... never misses a power-saving opportunity, no matter how small. It even regulates the processor between keystrokes, reducing power between the letters you type.*

Figure 1.1a shows the accompanying graphic. Microsoft's "Building Windows 8" blog also addresses OS power efficiency [69], referring to power management as

> *... a core OS capability that is critical on any chip architecture and any PC form factor.*

Abundant prior work has identified power consumption as a source of information about a system's internal state [7, 10, 19, 39, 42, 44], and thus a side channel that can be used for attacks, with a focus on simple embedded systems or attacks based on

targeting a single power-consuming subsystem. A reasonable question in light of this past work is what the impact of increasing energy proportionality is on the effectiveness of power side channels at a coarser, system-level granularity. Prior work has not considered the analysis of general-purpose computing devices at such a granularity.

1.2 Thesis Statement and Summary

The motivating idea of this thesis is:

> *The increasing energy proportionality of commodity computing devices has created a system-level power side channel that enables both new attacks and defensive techniques.*

Because the trend toward energy proportionality is not a new phenomenon, widely deployed computer systems are already amenable to system-level power analysis. Examining hardware available on the market today, this thesis identifies 1) malicious applications of the system-level power side channel, 2) constructive applications of the system-level power side channel, and 3) which system components consume power in the most characteristic ways.

1. **Malicious applications**

   Like many side channels studied in previous work, system-level power consumption has the potential to leak private information about a computer or its operator. To demonstrate the potential for *fine-grained* inferences about system workload, I describe and demonstrate a supervised learning technique capable of identifying which webpage from a training set a computer is loading with high accuracy. In particular, I evaluate this classifier with a variety of perturbations in measurement conditions to find the limits of its approach to classification.

2. **Constructive applications**

Taking a less traditional view of power analysis, I describe and evaluate a system for detecting abnormal behavior on embedded devices via power analysis. A key question for system and network administrators is whether they can detect intrusions without compromising these machines' operation or voiding warranties. System-level power analysis offers promise as one technique to address this problem. I take the examples of industrial control systems and medical devices to illustrate how aberrant operating conditions can be reliably detected via power analysis.

3. **Modeling and causative analysis**

   Finally, I characterize how different hardware components and workloads impact the efficacy of system-level power analysis. For some simple embedded systems, this characterization allows direct attribution of almost all changes in power consumption over time. For highly integrated commodity hardware, characterization relies on empirical correlation measures under a variety of workloads. This work sheds light on why the machine-learning based classifiers are effective for different applications.

1.3 Contributions

In the course of investigating the effects of energy proportionality on security and privacy, this thesis makes the following high-level contributions:

- The description and analysis of an attack on privacy that identifies specific web-browsing activities via an AC power side channel.

- PowerTrip, a nonintrusive anomaly detection system for safety-critical embedded systems that offers protection to devices that are historically difficult to protect with software- or network-based approaches.

- An investigation of the underlying causes of information leakage via system-level power consumption that elucidates under what conditions classification is effective.

By demonstrating that system-level power analysis is an effective means of making detailed inferences about a computer's internal state and investigating the underlying causes, this thesis will help to predict how ever-increasing energy efficiency is likely to impact future applications of system-level power analysis. Ideally, this thesis will help future system designers to understand the risks of malicious power analysis, how it can be defended against and how constructive power analysis can be fruitfully applied.

1.4 Thesis Outline

The remainder of this thesis is organized according to the claimed contributions as follows.

Chapter 2 provides shared background on the topic of whole-system power analysis — describing how power is consumed in the systems considered in this thesis and how I gather power traces for analysis.

Chapter 3 addresses power analysis attacks. It analyzes an attack that identifies webpages using power traces from commodity computers.

Chapter 4 considers constructive applications for system-level power analysis — presenting a nonintrusive anomaly detection system for embedded devices.

Chapter 5 characterizes the hardware components and workloads that contribute most to the efficacy of system-level power analysis.

Chapter 6 discusses the implications of this work to future applications of system-level power analysis and identifies how continued improvements in energy proportionality may affect security and privacy.

CHAPTER 2

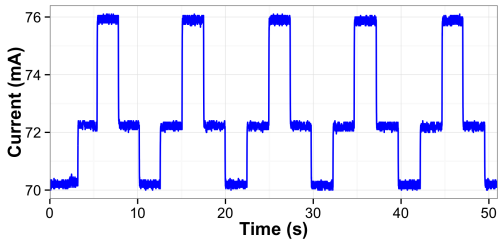
BACKGROUND IN SYSTEM-LEVEL POWER ANALYSIS

This chapter provides background on power analysis, with a focus on the techniques used in this thesis. It defines system-level power analysis, distinguishes alternating current power from direct current power, summarizes how empirical traces of each differ from idealized models, and describes the trace capture techniques used in this work.

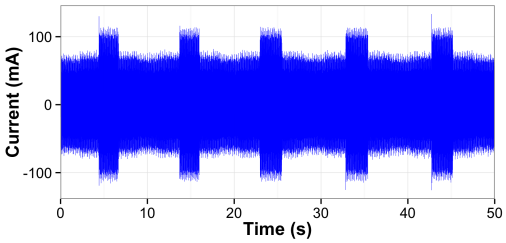
2.1 Power Analysis, Transmission, and Consumption

Power analysis is the process of making inferences from changes in power consumption over time [44]. System-level power analysis applies specifically to power traces gathered at the granularity of a whole device, i.e., at the "plug" connecting the device to the power infrastructure. Any system-level power trace of a computer necessarily includes contributions from every component that draws power, such as CPU, GPU, disk, and memory—which incorporates more information than any individual component's power consumption, but also conflates the signals from these subsystems, destroying some fine-grained information. Given a power trace, one would like to know what information it contains, and how to extract that information. In general, the techniques best-suited to alternating current (AC) and direct current (DC) power are not the same.

Power grids transport electricity to consumers using AC, in which the electric charge periodically reverses direction. Many devices consume AC power directly, but computing devices operate on DC power. Large industrial installations that comprise



(a) A DC trace of a programmable logic controller (PLC) periodically switching among four states. All current values are positive, with some ripple evident when the PLC remains in the same state. The values close to 72 mA represent one of two internal states that alias in the trace.



(b) An AC trace of a PLC switching among the same four states as in Figure 2.1a. The constant 60 Hz oscillations obfuscate the underlying signal and make it difficult to discern more than two states with the naked eye.

Figure 2.1

many DC-powered devices may convert AC power to DC power at a centralized location and distribute DC to devices within the facility. Consumer-oriented and commodity hardware, such as personal computers, draw power from standard outlets and handle power conversion themselves. This thesis considers computers used in industrial control, medical, and consumer contexts, so we consider both AC and DC devices.

To convert AC to DC power, computers use switched-mode power supplies (SMPSes) [29]. SMPSes take AC power as input (at 120 V in the United States) and provide DC power as output at a variety of voltage levels that computer subsystems consume directly. In the case of notebook computers, the power supply takes the form of a "brick" at some point in the power cord that provides a single DC voltage to the computer. Desktop form-factor computers typically include the power supply in the chassis and provide multiple DC voltage levels.

Power analysis requires power traces as input. A power trace is a sequence of $\langle time,\ power \rangle$ pairs [44]. The interpretation of changes in the power values over time depends on whether the trace represents AC or DC power, and the measurement point from which the trace was gathered.

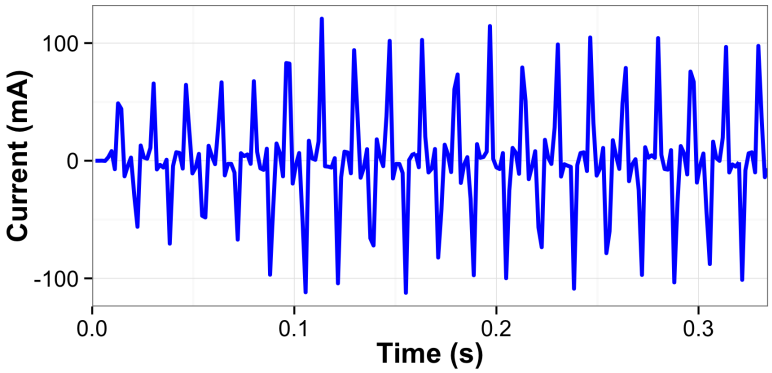


Figure 2.2: A closer look at an AC power trace that shows how heavily the SMPS in the load and other noise from the grid distort the sinusoidal carrier.

DC traces reflect changes in power consumption as increases and decreases in the sampled wattage over time. In the absence of changes in consumption, a DC power trace appears approximately as a line with slope zero. In practice, fast-moving changes in power consumption due to irregularities in consumption create noise even during "stable" periods, as Figure 2.1a shows.

AC power traces are more difficult to interpret. The periodic changes in charge direction manifest as a continuous sinusoidal fluctuation symmetric about zero (at a frequency of 60 Hz in the United States). Changes in power consumption are reflected as changes in the amplitude of the sine wave, with higher amplitude corresponding to more power consumption. In an ideal model, the power consumption one would observe in an equivalent DC trace is simply amplitude-modulated onto the sinusoidal carrier. Unlike DC power, however, the inefficiency of the SMPS in the device or *load* consuming AC power creates additional artifacts in power traces that complicate analysis, as Figures 2.1b and 2.2 illustrate.

2.2 Capturing Power Traces

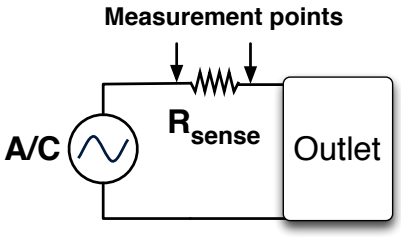
There are a variety of options for capturing power traces. This work focuses on a single method that applies to any device drawing power via a wire and does not

require modifications to the device under test. We placed a sense resistor, a single compact circuit element, in series with the devices we monitored, as depicted in Figure 2.3a. As power passes through the resistor, the voltage delivered by the source decreases based on the rated resistance. According to Ohm's law [58], which states that $I = \frac{R}{V}$ where I is the current in amperes, R is the rated resistance in ohms, and V is the observed voltage across the resistor in volts, we can calculate the current consumed by the load. After calculating the current consumed by the load, we can solve for power using the relationship $P = VI$, where P is the power in watts, V is the voltage from by the source, and I is the calculated current in amperes. We use the terms *power trace*, *voltage trace*, and *current trace* interchangeably because they are all directly proportional to one another in the context of our measurements.

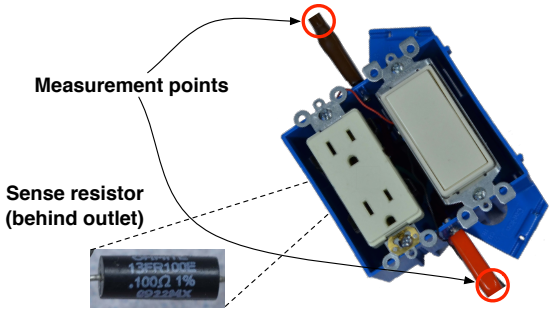
The measurement strategy varies slightly with the device under test. For DC devices, the sense resistor can be placed in series with either the positive or negative terminal delivering power to the device. DC devices — which do not include SMPSes — generally provide an accessible interface for power delivery, but it may take different forms for different devices. AC devices have the benefit of standard wiring interfaces, so we instrumented a common North American NEMA 5-15 outlet. Modern AC outlets have three terminals: *hot*, *neutral*, and *ground*. To measure a device's instantaneous current on the circuit, we placed a sense resistor (0.1 Ω, 1% tolerance) in series with one terminal of an outlet (Figure 2.3b), into which we plugged the device under test. For ease of experimentation, we extended the outlet from the wall by stripping one end of an extension cord and plugging the other end into an uninstrumented lab outlet.

Safety note: This chapter is not a manual for electrical safety. Measuring "hot" terminals is potentially fatally dangerous and should be conducted only under qualified supervision. Do not try this in a non-laboratory setting.

To rapidly record and store power traces, we attached an Agilent U2356A data acquisition unit (DAQ) to the terminals of the sense resistor [3]. The DAQ samples the



(a) A circuit diagram of our measurement technique. The difference in voltage on either side of the resistor (R_{sense}) is proportional to the power consumed by the AC device at left. The same measurement technique applies to DC circuits.



(b) An instrumented AC outlet for capturing power traces. A data-acquisition unit connects to measurement points on either side of a 1 cm sense resistor.

Figure 2.3

voltage across its probes and sends the data via USB to another PC (not the computer being measured). For the experiments described later in this thesis, we recorded 16-bit samples at a rate of 250 kHz (i.e., 4 Mb/s) to capture workload artifacts occurring at up to 125 kHz.

Alternate techniques to measure power consumption include Hall effect sensors [64] or custom powerline interface devices like that built by Gupta et al. [63]. Sensors relying on the Hall effect inductively couple with the electromagnetic field created around a wire carrying current, and so do not need to be inserted in series. The drawbacks to Hall effect sensors are that they are generally less sensitive and that they may sense nearby electrical fields not produced by the load of interest. Powerline interfaces allow single-point monitoring of a household from any outlet but, because they are not in series with a load, mainly observe changes in SMPS switching frequency or transient interference produced by device on–off transitions [63].

2.2.1 Information in Power Traces

Extracting information from DC power traces is relatively straightforward because they are simple representations of the power consumption over time. The limiting factor for inferences from DC traces is how directly one is able to map different operations to different power profiles. In the ideal case of a perfectly energy-proportional computer, each operation or internal state would map to a distinct power profile, but in practice many states may alias to the same profile because, at the level of an entire system, many power-consuming components may contribute to the single power consumption value observable with our technique. Figure 2.1a shows an example of aliasing. DC power measurements explored in previous research require tracing individual components' power consumption on a circuit board to gather more detailed information [44]. Such techniques require access to internal hardware elements and do not meet our goal of *nonintrusiveness*, or avoiding hardware modification/disassembly.

AC power traces are more complicated signals that require more sophisticated classification strategies in general. The bulk of this thesis focuses on AC power analysis based on the devices we wish to analyze. Past research has identified three primary sources of information leakage from SMPSes that manifest themselves in a computing device's power consumption: current fluctuations, reactive power, and changes in switching speed.

- **Current fluctuations:** If any particular component conveys information via its power consumption, then that information *may* appear in a power trace, where its power consumption is reflected as an increase in amplitude — or it may be lost to destructive interference from other signals. Prior work on Nonintrusive Load Monitoring (NILM) seeks to infer the on/off state of appliances based on changes in current consumption at the granularity of a household [32, 31].
- **Reactive power** Unlike efficient resistive loads (e.g., incandescent lights), the capacitive and inductive components inside SMPSes distort the shape of the si-

nusoidal AC waveform, drawing the current and voltage waveforms out of phase and returning power to the source in the form of *reactive power*. The amount of reactive power varies with the load, leaking information about system-level power consumption onto the power line.

- **Changes in switching speed** Like any electrical device that switches on and off, an SMPS emits electromagnetic interference (EMI) that other devices can detect. The switching speed varies with the components' aggregate demand for power. To meet emissions standards (e.g., FCC Title 47 CFR Part 15 [25]), SMPSes contain inductors that filter out noise at frequencies above the voltage regulator's switching frequency. This EMI filtering does not prevent activity information from appearing on the AC power line, as Gupta et al.and Enev et al. demonstrate [63, 29, 19].

2.3 Summary

This chapter provided the background on power systems and power measurements necessary to understand the modeling and classification approaches in this thesis. It defined system-level power analysis, summarized the essential characteristics of DC and AC power traces, and described the trace capture techniques used in this work.

CHAPTER 3

MODELING POWER CONSUMPTION

The development and evaluation of an effective classifier for a given set of traces—whether those traces represent power consumption or some other phenomenon—depends on an accurate model of the input. This thesis addresses two broad classes of devices: embedded control systems based on firmwares, and general-purpose computers running commodity operating systems but dedicated to a specific application. Some firmware-based systems such as programmable logic controllers (PLCs) that contain few discrete components are amenable to some level of *generative* modeling that maps computing tasks directly to power consumption. Commodity platforms, on the other hand, exhibit more complicated hardware/software interactions and tremendous state spaces. The classifiers that we apply to commodity platforms thus rely on *discriminative* models—learning how to discriminate among power traces rather than understanding how the power traces are produced by the corresponding workloads.

This chapter examines the challenge of modeling power consumption for the devices considered in this thesis and addresses the question of *why* power traces are differentiable. It is not necessary to answer this question in the context of creating a classifier for a narrowly specified application, but answering the "why" question is important when reasoning about the limits of a classification approach or how changing conditions are likely to affect performance. To that end, this chapter examines the hardware and software variables that produce identifiable differences in power traces and the underlying causes of distinct power signatures on both embedded devices

such as PLCs and commodity computers that are somehow constrained to a limited state space. It answers the following questions:

- What are the major barriers to the development of detailed generative models?
- Which computer subsystems account for the most system-level power consumption?
- What types of software workloads produce the most easily identified power traces?

To answer these questions, we contrast generative and discriminative models for different pieces of hardware, present approximate power budgets for different computing devices, and quantify the relationship between system-level AC power consumption and each DC power channel provided by a commodity SMPS in a personal computer.

We rule out the RAM, hard disk, and GPU as key sources of information in system-level desktop power traces and identify the CPU as the most likely source. We also find that GPU-intensive workloads provide comparatively little information, and that more diverse workloads such as webpage rendering are easier to identify.

3.1 Generative and Discriminative Models

Generative models map specific processes to the observable phenomena they produce. They have the potential to produce highly accurate classifiers for which it is possible to reason about the precise limitations based on the model, but exhaustive modeling of even simple systems is a difficult task. For their work on the energy-aware Mementos system [65], Ransford et al. augmented a cycle-accurate simulator of the MSP430 family of microcontrollers [20] with an empirical energy model for the purposes of simulating power failures on batteryless devices. This detailed model

required knowledge of the instructions executed (extracted from compiled programs) and the differences in energy consumption for different classes of instruction, e.g., those addressing registers, RAM, flash, or peripherals like analog-digital converters. Creating such detailed models requires extensive measurements on a per-platform basis. This resource investment makes it an impractical approach for many of the platforms discussed in this thesis, since the goal is to rapidly produce a model based on a short profiling or learning period and without access to object code. Less detailed generative models, however, may still be valuable.

Discriminative models allow a classifier to identify outputs belonging to different classes, but do not necessarily provide any insight into what the sources of information are. For example a classifier trained on power traces of a lamp might learn that the lamp is on when the power consumption is high and off otherwise, but this model alone does not offer any insight into how a lamp works. For our purposes, we focus on discriminative models developed internally by a supervised learning algorithm. These models have the advantage of working with standard approaches and do not require many a priori assumptions or a detailed understanding of the underlying process. Understanding the limits of a classifier's internal model and predicting how it will perform under changing conditions, however, is a secondary problem that must be addressed when using discriminative models.

3.2 Building a Generative PLC Model

A PLC is a computer designed to issue electrical signals to other devices along control lines, according to some program its operator has provided. PLCs are ubiquitous in industrial and commercial automation settings. PLC firmware programs are structured as loops, each iteration of which encodes I/O or computation.

Some PLCs run embedded operating systems such as real-time Linux variants, but these are relatively recent arrivals; traditionally PLCs run a single firmware program

Figure 3.1: A Siemens S7-1200 1214C AC/DC/Rly PLC (top) with a "trainer" box providing input and output hardware (bottom).

without a multitasking OS underneath. A typical PLC of either type comprises a power supply, a CPU, and I/O modules that contain solid-state relays, transistors, and digital–analog conversion components. After an operator loads compiled firmware onto the PLC via a temporary wired connection, the firmware program runs as soon as the PLC is powered.

PLCs, which we consider in depth in Chapter 5, are amenable to generative modeling because their firmwares generally map a small set of inputs to a small set of outputs. The Siemens S7-1200 PLCs we focus on in this work, for example, are equipped with a microcontroller, 4 inputs, 4 outputs, and pulse-width modulation output support. Figure 3.1 depicts one of these PLCs. A detailed generative model in the style of Mementos [65] would profile the run time energy costs of each instruction or class of instructions and produce a series of expected ⟨time, power⟩ pairs.

This type of modeling proves to be difficult and of limited value when applied to PLCs for a number of reasons. First, the object and binary file formats used with Siemens PLCs are not open or well-documented and neither are the performance details or instruction set architecture for the MCU. Second, the PLCs implement a visual programming model that maps to a finite state machine. In the absence of input changes or logic dependent upon a clock signal, the PLC will remain in a steady state with no significant power consumption changes. Finally, many PLC input and output circuits are identical in terms of hardware, comprising a control pin from the MCU and a solid-state relay or transistor to electrically isolate the PLC from its I/O devices.

Work by Cárdenas et al. addresses these limitations by directly monitoring input and output hardware as a way of enforcing the set of valid I/O states [11]. Their system does not attempt to model the MCU's internal state. The work we present in Chapter 5 takes a similar approach to DC-powered PLCs, but seeks generality across deployments by monitoring only the PLC itself without knowledge of the particular I/O hardware. Using our approach, we find that we can reliably identify the *number* of inputs and the *number* of outputs in a "high" state, but cannot differentiate *which* inputs and outputs are in a high state. The classifier is thus limited to determining whether power traces are plausible based on the number of inputs and outputs expected. As the number of inputs and outputs increases, the potential for aliasing increases. This is because the total number of distinct states is proportional to 2^n, but the total number of distinguishable I/O states is proportional to n^2 where n is the number of inputs and outputs.

The advantages of this model are that the limitations are readily apparent and the inputs are well-understood. We can also extend this model to AC-powered PLCs given the assumption that state changes occur at a frequency significantly less than 60 Hz, as Chapter 5 shows. A future classifier based on the model could be designed

Condition	Power (W) vs. Baseline
Idle Baseline	4.7
Input high	+0.3
Output high	+2.2

Table 3.1: The approximate power budgets for an S7-1200 PLC. Numbers beginning with + are relative to the baseline of 4.7 W.

to work even without an online training period. For our PLCs, hardware profiling proves to be a simple task. Using a P3 Kill A Watt power monitor [62], we recorded the baseline power consumption and the changes in power consumption for input and output hardware. The results are summarized in Table 3.1. With hardware power budgets and knowledge of the state space, one could predict the power consumption levels that a given firmware has the potential to produce.

3.3 Building Discriminative Models for Commodity PCs

Commodity PCs represent a formidable modeling challenge because of their hardware and software complexity. CPUs and GPUs each include billions of transistors and operating systems consist of tens of millions of lines of code [56]. On top of this complexity, features like multi-threading and interrupt handling create an environment in which it is unclear how to even precisely describe a single distinct state. Rather than attempting to tackle this problem directly, side-channel research, including Chapters 4 and 5, generally relies on discriminative models or explicitly models only a portion of a complex system [7, 19, 29, 42, 48, 61, 53].

The general form of a discriminative model for supervised learning is a set of classifier parameters learned from the training set. For the random forest classifier used by PowerTrip, these parameters are feature values assigned to nodes in a set of decision trees. Under the assumptions that the training and testing sets are drawn

from approximately the same statistical distribution and that the feature vector contains enough information to differentiate the classes, this model allows the classifier to reliably label inputs correctly. A difficult question to answer is: under what conditions are these assumptions invalid? Finding the answer to this question is key when reasoning about changing classification conditions.

Chapter 4 explores many variations on the webpage classification problem with the goal of understanding when the testing set is different enough from the training set to degrade classifier performance. We found that most changes in browsing conditions had little effect on accuracy, but hardware and software changes largely invalidated the learned model. To avoid repeated classifier training and testing, another approach is to isolate the sources of identifiability by finding other characterizations of power consumption.

3.3.1 Understanding sources of identifiability

To identify promising sources of information for system-level power analysis, the components consuming the most power are a possible starting point. We measured the power consumption of a 2008 MacBook under a variety of workloads designed to stress individual subsystems. We used a P3 Kill A Watt power monitor [62] to measure power consumption. Table 3.2 summarizes the results, which suggest that the MacBook's CPU and GPU dominate power consumption under load. The network interfaces and solid-state storage draw comparatively little power.

These data suggest that, if each subsystem exhibits the same degree of energy proportionality and is placed under the same load, the CPU and GPU contributions to power consumption are the most visible in power traces and also contain the most information. Whether these two assumptions are true in practice is difficult to assess.

Prior side-channel work has leveraged network characteristics such as packet timings [68] or lengths [82, 83] to classify webpages according to their network traffic. A

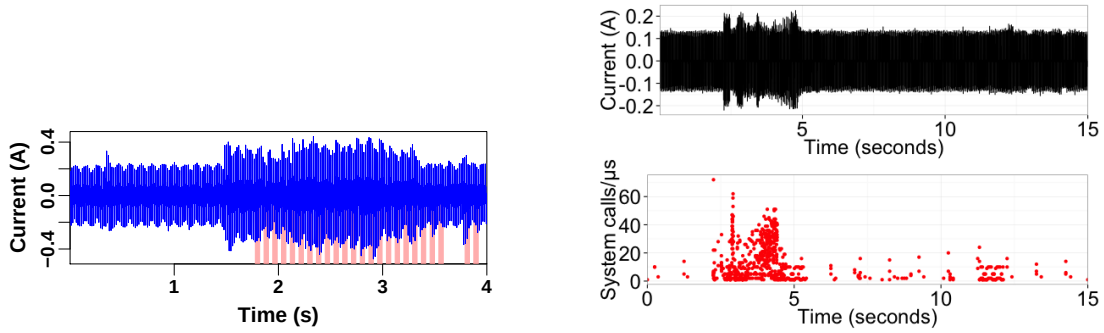
Condition	Power (W) vs. Baseline
Baseline (idle, screen off)	**8**
One core at 100%	+7
Two cores at 100%	+11
GPU at 100%	+11
Wired network saturated	+2
Wireless network saturated	+3
File copy, SSD to SSD	+6
Screen at maximum brightness	+6

Table 3.2: MacBook power consumption under various types of load. Numbers beginning with + are relative to the baseline of 8 W.

natural question to ask is whether power traces provide information about network characteristics based on the power that network components consume. An experiment suggests that system-level power traces in fact do not map exactly onto network traffic. We tapped the activity LED of a network switch port to capture a representation of a computer's network traffic while also tracing the computer's AC power line. Figure 3.2a shows an example from our tests. The computer consumes power performing other tasks before the network interface actually begins to send and receive packets. Furthermore, the AC power provides insight into client scripts and rendering loads unavailable in a network trace.

Power consumption appears to be more strongly correlated with system calls than with network activity as shown by Figure 3.2b. Tracking the number of system calls initiated by the browser process with DTrace captures memory allocation and disk I/O in addition to network activity, enabling monitoring of all of the components we have identified as major power consumers.

While these experiments point to the CPU as a likely primary source of information, they are not conclusive because they do not allow us to attribute the power consumption for a given application to a specific subsystem.



(a) The network activity is correlated with high current consumption, but is not the only cause. Spikes before and after network activity (which is depicted in red) show that local computation dominates the consumption.

(b) The system call activity (as measured by DTrace) is also correlated with high current consumption, and our results suggest that systems exercised by system calls are a major cause of consumption.

Figure 3.2: Time-domain plots as a MacBook loads webpages. Both network activity and system calls appear to correlate with energy consumption.

3.3.2 Attributing System-level Consumption

Directly discovering which components account for the majority of the system-level power consumption requires that we directly measure the various subsystems. Neither desktops nor laptop computers are designed to allow this type of profiling. The SMPS in a desktop computer is integrated into the chassis and does not expose any of the DC power channels that it provides to the internal components. Laptop computers expose their SMPSes, which take the form of power cord "bricks", but the DC side of the SMPS only provides a single voltage. Sampling this DC signal does not provide more detailed per-component information, as Figure 3.3 shows.

We built a piece of custom measurement hardware to gain per-component visibility. Extending the basic technique of placing a sense resistor in series with the load, we instrumented each wire carrying DC power from a desktop SMPS to the system components. This includes power cables for the hard disk (HDD) and GPU, as well as multiple cables carrying various voltages to the motherboard. Figure 3.4 depicts a rough schematic of the measurement harness and Figure 3.5 shows the harness itself attached to our test machine, a Dell Vostro desktop with quad-core Intel Core

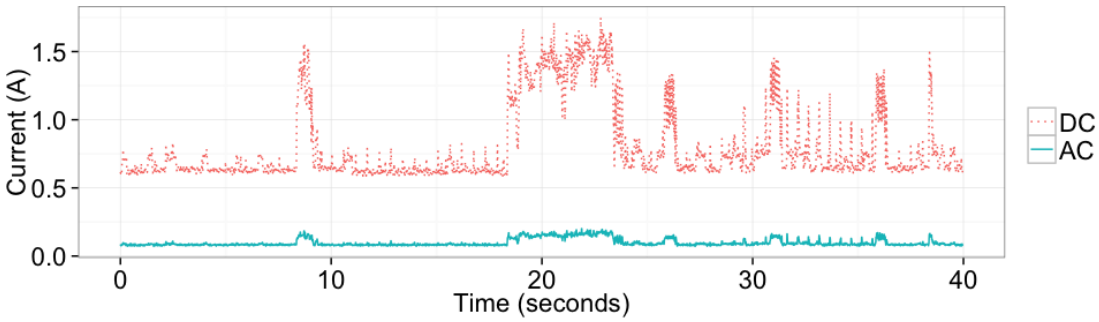


Figure 3.3: A time-series plot of the simultaneous AC and DC current consumption while loading a webpage. The DC current is plotted as the moving average with a red dotted line. We plot the AC current as its envelope for easy comparison. The AC envelope closely matches the DC consumption, which is also a system-level power channel.

i5 processor, 4 GB of RAM, AMD Radeon 6450 GPU, and 250 GB SATA magnetic drive, tested under Windows 7. Finally, we sampled the voltage drop across all of the resistors simultaneously using an Agilent U2356A Data Acquisition Unit (DAQ) [3]. The DAQ output is a set of simultaneous power traces representing the system-level consumption and the consumption of each instrumented DC channel.

Identifying which channels provide power to which components is not straightforward. Power connectors that go directly from the SMPS to a component like the hard disk or GPU are unambiguous, but others are not. Without schematics of the motherboard's power distribution circuitry, we must make educated guesses about where it routes power based on a set of workloads designed to stress specific subsystems.

For analysis, we calculate the mutual information between the AC power trace and each of the DC channels (labeled as DC1, DC2,...,DC28) to attribute the system-level power consumption to the various subsystems. The mutual information is a measure of the statistical dependence between two random variables, calculated as the difference between the marginal unconditional entropy of one variable and the entropy of that variable conditioned on the other, expressed as: $I(feature; class) = H(feature) - H(feature|class)$ where H is the entropy and I the mutual information.

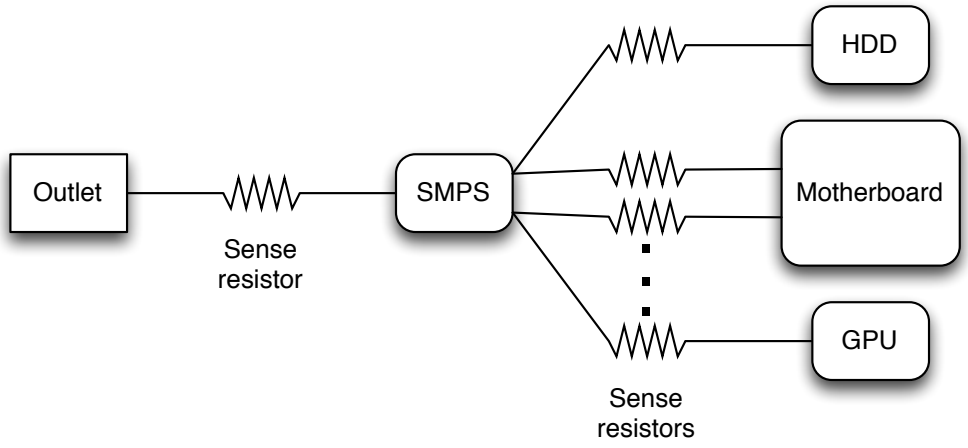


Figure 3.4: An approximate schematic of the measurement harness we used to correlate AC and DC power consumption. A sense resistor placed in series with each wire coming from the SMPS allows us to simultaneously sample the system-level and per-component power consumption.

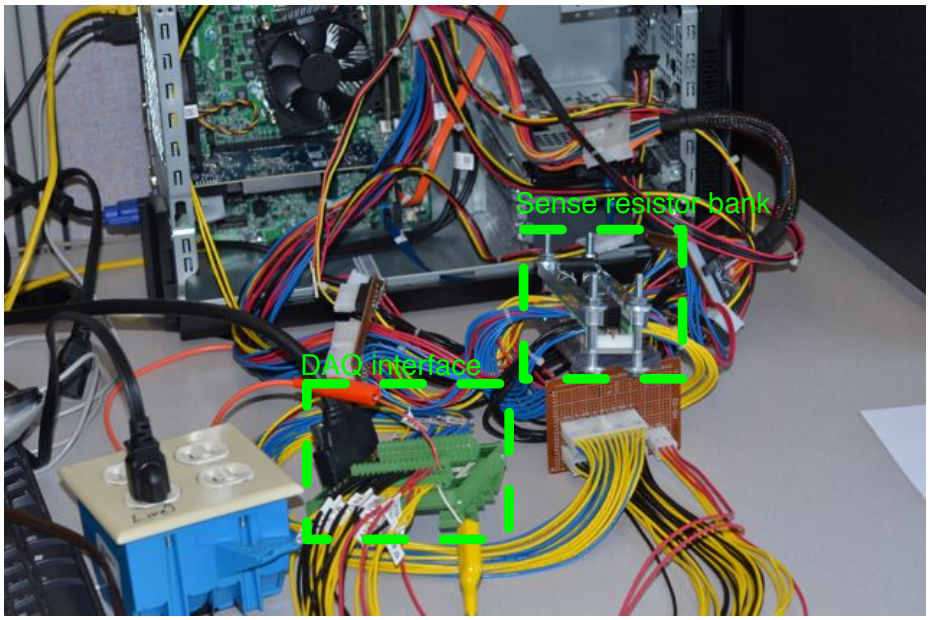


Figure 3.5: The wiring harness we used to correlate AC and DC power consumption. The bank of sense resistors and the DAQ interface are outlined in green.

Workload	Top channels	Max Mutual Info
RAM test	1, 21	0.17
1/2/3/4 infinite loop threads	21, 22	0.29
Large file copy	26, 22, 25, 21	0.26
OpenGL demo	None	0.08
Idle	25, 22, 21	0.19
1080p video	None	0.05
Static webpage load	25, 26, 22, 21, 20, ...	0.70
`cnn.com` load	25, 26, 22, 21, 20, ...	0.32
Emulated malware infection	26, 25	0.35
"Ramnit" malware infection	26, 25	0.29

Table 3.3: The workloads for which we gathered simultaneous AC and DC traces, and the results. The workloads listed in the top half of the table attempt to create a best-case analysis scenario for a specific subsystem. Those in the bottom half represent real-world applications. The top channels are those for which the normalized mutual information exceeded 10% of the AC channel.

We calculate the mutual information in bits and then normalize it, with a value of 1 being equal to the mutual information between the AC channel and itself (which is the entropy of the AC channel). This metric is the *normalized mutual information*.

Changes in the normalized mutual information across different workloads indicate which components are best represented in the system-level power consumption. We tested a number of workloads designed to stress individual subsystems and also to mimic the malware and webpage workloads explored in this thesis. Table 3.3 lists the workloads tested. We only consider channels to carry a significant amount of information if the normalized mutual information exceeds 0.10.

The results summarized in Table 3.3 allow us to come to several conclusions, which we present in terms of components and then in terms of workloads.

3.3.2.1 Component results

The RAM test is the only workload for which DC1 appears as a top channel. DC1 is the 12 V rail that goes directly from the power supply to the DVD drive. This is unsurprising, because the test runs from a bootable CD. The other top channel,

DC21, appears in most of the other workloads. It is one of the 12 V rails going to the 24-pin main motherboard connector [35, 36]. These observations lead us to conclude that the RAM's power consumption provides little information because it does not change the order of top channels when we stress the RAM almost exclusively.

The GPU also provides little information. Like many discrete (stand-alone) graphics cards, the AMD Radeon HD6450 in our test system has a dedicated power connector, which never appears in the top channels even when running an OpenGL demo that should saturate GPU resources. The GPU also draws power from its PCI Express connector, but the two GPU-intensive workloads yielded *no* DC channels that met our normalized mutual information threshold.

Similarly, the hard drive provides little information about system-level power consumption. DC2 and DC14 are the two power rails dedicated to the hard drive and neither appears as a top channel for any workload—including a large file copy.

Finally, the CPU appears to be a major source of information. The infinite loop workload yields two top channels, DC21 and DC22, which each appear in many of the other workloads. These channels power the 24-pin motherboard connector, but their relative value in other traces suggests that they route to the CPU.

3.3.2.2 Workload results

The top channels for the real-world workloads are largely the same, and are all part of the 24-pin ATX connector on the motherboard. The maximum mutual information observed in the real-world workloads is generally higher than it is for individual subcomponent tests. Without more information about how the motherboard routes each DC channel, we cannot conclude for certain which components actually provide the most information though discovering this information would provide a detailed mapping [36, 35].

The real-world workload experiments do show that workloads placing loads on several components provide the most sources of information. The webpage tests both make use of the CPU, RAM, HDD, and network interface and each has more than five channels meeting our normalized mutual information threshold.

3.4 Summary

Developing a power consumption model is a necessary step in the development of a power trace classifier. For relatively simple embedded systems like PLCs, a generative model based on empirical measurements is sufficient to design an effective classifier with known strengths and weaknesses. The complexity of commodity computing hardware and software systems makes building a comprehensible generative model prohibitively difficult. Instead, a suitable feature vector combined with a general-purpose classifier allows the creation of an accurate discriminative model. The major drawback to a discriminative model is the difficulty inherent in predicting under what conditions the model will break down.

This chapter summarizes the specific barriers to constructive a generative model for a PLC and also for commodity computers and shows how to build upon a discriminative model of a complex system to profile the underlying reasons *why* a classification approach is successful. We rule out the RAM, hard disk, and GPU as key sources of information in system-level desktop power traces and identify the CPU as the most likely source. We also find that GPU-intensive workloads provide comparatively little information, and that more diverse workloads such as webpage rendering are easier to identify.

CHAPTER 4

POWER ANALYSIS ATTACKS

Power consumption is well-established as an undesirable side channel that poses privacy risks, including the disclosure of user inputs, cryptographic keys, and display contents [77, 78, 44, 28, 48, 19]. The ever-increasing energy proportionality of commodity computing hardware according to Koomey's Law (see Chapter 2) has the potential to exacerbate long-standing concerns about power side channels. As power consumption scales more closely with workload, a key question is whether new side channels arise or existing side channels become more accessible or leak more information.

This chapter explores the impact of systemwide power analysis on computing privacy, presenting a new attack against user privacy based on monitoring AC power consumption at the plug. It answers the following questions:

- To what extent can we identify web traffic from AC power traces?
- What characterizes the information leaked by a modern computer via AC power consumption?
- To what changes in measurement conditions are AC power signatures robust?

This chapter demonstrates that relatively coarse-grained measurements in combination with standard machine learning techniques are sufficient to glean fine-grained information from a commodity computer. The major contribution is the examination of how accurately an attacker can distinguish which webpage a user is loading under

a wide range of conditions. Our classifier can identify which of 50 candidate webpages a browser is loading with 99% precision and 99% recall. The classifier is also robust to several changes in browsing conditions. While this chapter focuses on the problem of identifying webpages, the attack's accuracy and robustness have implications for the privacy of other potentially sensitive information that may become manifest in power traces.

4.1 Introduction

Computer users commonly assume that software mechanisms, such as in-browser encryption, protect their private information. Research on side channels has challenged this assumption by showing that computer components such as the CPU [44] and the keyboard [78] can leak private information. Along the same lines, this chapter examines the feasibility of inferring private information from a general-purpose computer's AC power consumption, despite significant additive noise from the power grid [14].

Past work has exploited AC power side channels for information leakage, but at the level of an entire household [60] or a device with a constrained state space [19, 14]. For example, a television that is dedicated to displaying videos produces consistent power consumption over multiple plays of the same video because there is a direct relationship between total screen brightness and power consumption per frame. Given a small number of candidate videos, it is possible to identify which of them is playing [19]. A general-purpose computer, on the other hand, exhibits a tremendous state space because of its practically unconstrained operation. Executing the same computing task at different times may result in different power consumption patterns because of different background tasks or I/O workloads (e.g., network activity). Nevertheless, we find that system-wide traces of AC power consumption leak enough information about the operation of a general-purpose computer to identify the webpage that the

computer is loading (out of a set of known pages). Because browsers use a diverse subset of the available hardware components, our results suggest that this technique may generalize to other computing workloads.

Several factors work to our advantage. Web browsers increasingly take advantage of hardware to improve the user's experience (by, e.g., executing native code [84]), resulting in resource consumption that scales with the webpage's complexity. Another factor is that modern computers and operating systems aggressively try to reduce power consumption [73], resulting in energy-proportional computing in which the power consumption tightly fits the workload [14]. Both of these factors increase the system's dynamic range, which in turn increases the information available in power traces.

There are also challenges to the task of identifying webpages from AC power consumption. The fundamental challenge is separating interesting activity from baseline power consumption, which a computer's power supply aggregates. Other challenges stem from the dynamic nature of the Internet and modern websites. The round trip time for fetching webpages may change over time; many websites include scripts that run long after the page loads, and many customize content for each visitor.

This chapter's contribution is the analysis of an attack on privacy that identifies specific web-browsing activities via an AC power side channel. We characterize and measure this side channel by designing methods to extract patterns of power consumption as a computer loads a webpage. These patterns, which are obscure in the time domain but more apparent in the frequency domain, act as power signatures that allow an eavesdropper to determine which webpage is being loaded. Additionally, these power signatures are robust against a variety of changes to the computing environment, including background processes, changes in network location, the use of a VPN, or even the use of a different computer. Because most of the identifiable information occurs at under 10 kHz in the frequency domain, capturing and exfiltrat-

ing power measurements is within reason for a simple embedded device that could fit discreetly inside a power outlet.

Using a covertly modified electrical outlet to record power consumption, we trained and tested a classifier with over 100 hours of traces representing more than 13,000 page loads from a set of 51 webpages from sites representing over 30% of global page views. Given a power trace with 51 possible labels, the classifier identified the correct match from the training set with 99% precision (resistance to false positives) and 99% recall (resistance to false negatives). Given an unlabeled trace from one of 441 webpages *not* in the training set, the classifier's false positive rate is less than 2%. In some cases, the classifier succumbs to overfitting when trained on traces from a single computer, confusing other power activity for browsing activity. However, when trained on traces from two computers, the classifier identifies the common patterns that are due to browser activity and can correctly label unseen traces from either computer.

This work conceptually bridges the gap between previous work on circuit-level direct-current (DC) power analysis [44] and coarser-grained, household-level activity recognition via AC power measurements [32, 63, 29]. This chapter also proposes several hardware and software countermeasures to minimize information leakage.

4.1.1 Threat model

Focusing on the AC power side channel, this chapter considers an attacker with physical access to a power outlet the victim might use. Possible easy targets include outlets in coffee shops and airports. A simple modification to a power outlet enables discreet data recording. Because most users implicitly trust power outlets, an attacker may gain easy, *persistent* access to a victim's power-consumption patterns.

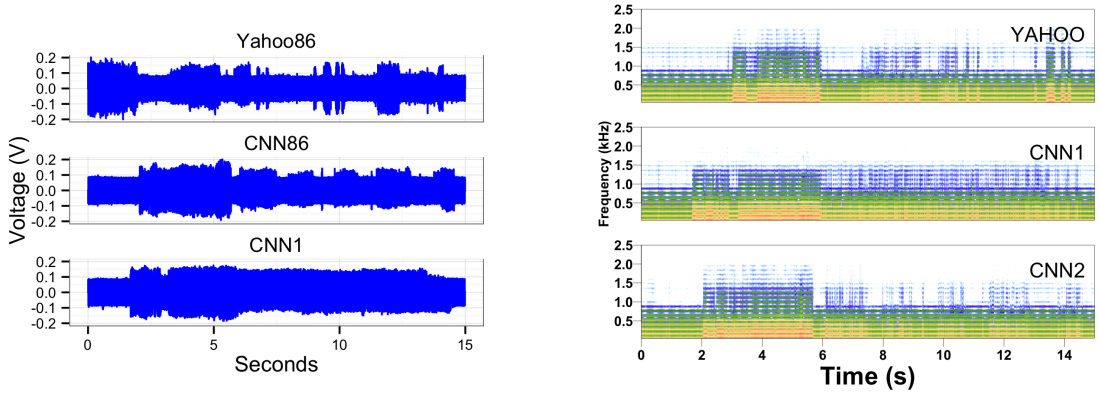
4.2 Background and Challenges

This section distinguishes AC power measurements from DC power measurements, which have been studied extensively in literature about side channels [44, 10]. It breaks down the power budget of a computer and discusses how the actions of a web browser influence power consumption. Finally, it explains some of the challenges inherent in our attempts to classify AC power traces.

4.2.1 AC Versus DC Power Traces

DC power measurements explored in previous research require tracing individual components' power consumption on a circuit board. Such techniques require access to internal hardware elements and are overly intrusive from the viewpoint of our threat model. An attacker conforming to our threat model seeks a power-analysis technique that is relatively nonintrusive and does not involve opening or modifying the victim's computer. AC power is, in a sense, the least common denominator: every computer operates on power drawn from a power grid. A laptop user may avoid drawing power from the grid by relying on the battery, but this is a temporary solution. Monitoring AC power consumption affords a *system-wide* view of the computer's activity.

The periodicity inherent in AC signals shapes our analysis approach. Whereas DC signals often feature prominent level shifts and other artifacts that are amenable to time-domain analysis, the *alternating* nature of AC current essentially convolves sets of sinusoids, making time-domain analysis difficult. The constant 60 Hz oscillations are particularly problematic because even minor phase misalignments foil many similarity metrics. We therefore perform analysis in the frequency domain. Before classification, we transform traces into the frequency domain (Figure 4.1b) using the Fourier transform. In addition to making certain signals easier to identify (e.g., 60 Hz utility power in the U.S.), this approach enables meaningful comparisons despite misaligned traces, or traces of different lengths, and the additive nature of the



(a) Time-domain plots

(b) Frequency-domain plots

Figure 4.1: Time- and frequency-domain plots of several power traces as a MacBook loads two different pages. In the frequency domain, brighter colors represent more energy at a given frequency. Despite the lack of obviously characteristic information in the time domain, the classifier correctly identifies all of the above traces.

SMPS's power consumption preserves frequency information from individual components' power consumption.

4.3 Approach: Supervised Learning Classifier

To distinguish among webpages, we adopt a supervised learning approach, in which we train a classifier on labeled AC power traces and then attempt to match unlabeled traces. An AC power trace contains artifacts of every powered computer component, each of which may have its own clock rate or power signature, and each of which processes information differently. We assume that disentangling these signals (multicore CPU, multicore video card, multiple drives, etc.) from a single AC power trace is intractable with current techniques, and instead focus on coarser-grained, *system-level* questions, such as which popular webpage the user is loading. It is prohibitively difficult to create a generative model of how specific computing tasks will map to power consumption. For this reason, a discriminative model — one that can discriminate among power traces rather than understand how traces are generated from hardware — is more appropriate. Our supervised-learning approach fits this

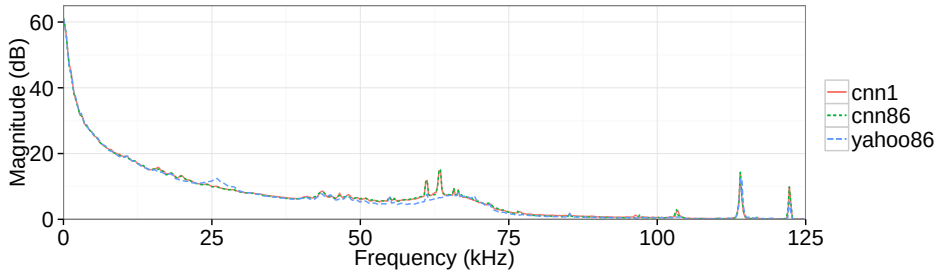
requirement; it requires only a labeled training set to build a model. Specifically, we train *support vector machines* (SVMs) using the open-source library libsvm [13].

4.3.1 Feature selection

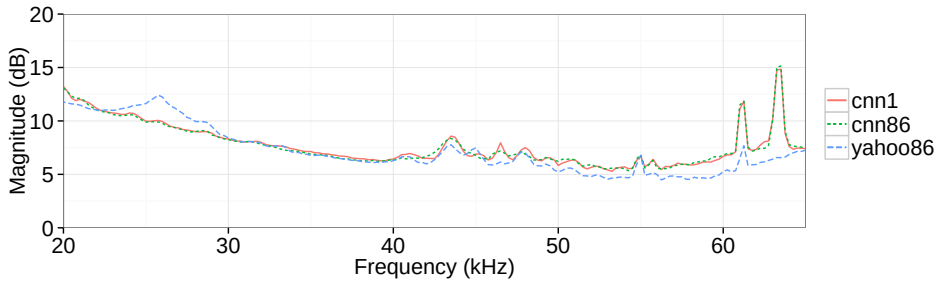
Classification requires extracting feature vectors on which to train a classifier. A naïve classifier might simply consider the *length* feature of a sample in the time domain, defined as the length of time for which power consumption remains above a predetermined threshold. However, background tasks add confounding noise in the time domain, obscuring the true endpoints of a specific task, and tasks often include periods of both high and low power consumption. A more robust approach is to classify traces based on features from the frequency domain.

We transform traces into the frequency domain by first calculating the spectrogram using rectangular sliding windows 1000 samples wide with 50% overlap. Rectangular windows are the simplest choice and 50% overlap is standard in many applications. We then collapse the spectrogram into a single Fourier transform by summing over all of the time steps. As a base set of features for classification, we divide the Fourier transform of each power trace into 500 segments, each 250 Hz wide, starting at 0–250 Hz and ending at 124.75–125 kHz, half the 250 kHz sample rate at which we recorded traces. This process yields 500 features, each of which represents the power present within one 250 Hz slice of spectrum over the duration of the trace.

As described in Section 4.2.1, classifying in the frequency domain allows meaningful comparisons between misaligned traces or traces of different lengths. Using the output directly from the Fourier transform as a feature vector is a simple approach and yields excellent results. It is also straightforward to visualize differences in the Fourier transform, as shown in Figure 4.2. Plotting the feature vector reveals consistent features between the two traces of `cnn.com` and recognizable differences between a trace of `yahoo.com` and the `cnn.com` traces, which are not obvious in the



(a) The full frequency range of our feature vectors.



(b) A tighter view of the same plot focusing on the most easily distinguished window.

Figure 4.2: Plots of three of our Fourier transform feature vectors. While the pages are difficult to separate visually in the time domain, the two `cnn.com` samples are indistinguishable to the eye in the frequency domain, whereas `yahoo.com` diverges around 25 and 65 kHz.

time domain. The plots also reveal that the differences between the two pages appear at low frequencies and again at the power supply's switching frequency of ∼65 kHz.

4.3.2 Classification

We train a binary classifier for each page in our corpus. After training all 51 SVMs, we use each of them to classify test samples. A test sample is an *unlabeled* 500-dimensional feature vector, obtained in the same way as the training samples, that is *not* in the training set. Each SVM determines whether the test sample was an instance of the webpage it was trained to recognize. In the interest of simplicity, we do not implement a multi-class labeling solution in which all 51 SVMs collectively generate a single output, but there are a variety of well-studied techniques for this purpose and libsvm implements several of them [34].

There are three notable details of the training process. First, we jointly normalize feature values across all samples. This prevents features with large values (e.g., the low frequency elements) from dominating features with smaller values. Second, we use standard 10-fold cross-validation to avoid overfitting at training time. By repeatedly splitting the training set and retraining each time, the classifier avoids the possibility of biases in a small number of training examples producing a biased model. Finally, we use a radial basis function (RBF) kernel, as recommended by the authors of libsvm [13].

4.4 Methods and Metrics

To cover a diverse set of webpages representing typical Internet traffic, we chose 48 webpages drawn from Alexa's list of the top 500 websites [5], discarding duplicates and adult websites. Alexa is a web analytics company that tracks website popularity using a browser toolbar installed in millions of browsers worldwide. The list of Alexa top sites is based on a large but biased sample, but we are not aware of any freely available alternatives without obvious biases. By Alexa's estimates, the top 48 websites we used represent over 30% of global page views (Figure 4.3). We added the top Google result for "cheap Viagra" as an example of a potentially embarrassing (or malicious) page. To include a page that loads with negligible latency, we added two authors' department's home pages, a < 1 ms round trip from one of our measurement points, bringing the number of webpages in our training set to 51.

The Alexa rankings list *websites*, but it is more meaningful to collect traces of individual *webpages*. Each of our traces represents an automated load of the front page of one of the 51 websites. To record realistic power traces of user browsing, we used a custom Chrome extension (see Section 4.4.1) to collect at least 90 consecutive traces of each page. For webpages that require users to log in before displaying useful information, we logged in as the first author. We believe that our choice to consider

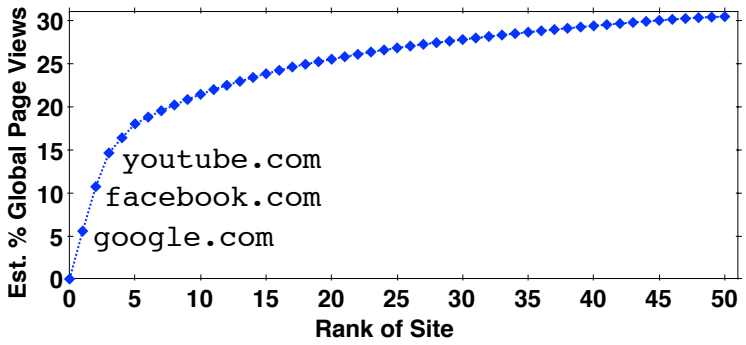


Figure 4.3: The top 50 websites according to Alexa in terms of percentage of global estimated page views.

front pages is reasonable because users are likely to visit the front page of a given website and then follow links to other pages. The notable exceptions to this tendency are bookmarked pages and direct links from other people or sites.

We evaluate our classifier's performance using standard metrics from machine learning. In the following definitions, *tp* and *tn* refer to true positives and true negatives (correct labelings), and *fp* and *fn* refer to false positives and false negatives (incorrect labelings).

Precision, $tp/(tp + fp)$, is the fraction of positively labeled examples whose labels are correct. It measures the classifier's ability to exclude negative examples.

Recall, $tp/(tp + fn)$, is the fraction of all the examples that *should* have been positively labeled that *are* correctly positively labeled. It measures the classifier's ability to identify positive examples.

We present experimental results in terms of *precision* and *recall* because the standard *accuracy* metric is potentially misleading. In most of our experiments, the number of negative examples is roughly 50 times the number of positive examples because, for each webpage, there are more traces of *other* webpages in the testing set than there are of that webpage. Because of this disparity between positive and negative examples, the classifier could achieve greater than 98% accuracy by simply

classifying all examples as negative. A perfect classifier would achieve 100% accuracy, 100% precision, and 100% recall. For any imperfect classifier, there is a tradeoff between precision and recall.

4.4.1 Experimental Setup

This section describes the experimental setup we used to capture AC power traces.

Using an instrumented outlet (described in Section 2.2), we measured the power consumption of two Apple MacBook computers, a Lenovo T410 laptop, and a Dell Vostro desktop PC, all running different operating system versions. Table 4.1 contains detailed hardware and software specifications. To approximate the environments of typical users, we used a stock installation of each operating system. In particular, we allowed default background processes to run. Experiments with the two MacBook computers were carried out approximately one year apart using similar but non-identical instrumented outlets.

Computers used in experiments
MacBook-1: 2008 model, dual-core Intel Core 2 Duo processor, 4 GB of RAM Intel GMA X3100 GPU, 80 GB Corsair SATA II MLC solid-state drive Mac OS 10.6.8. Battery removed.
MacBook-2: 2008 model, dual-core Intel Core 2 Duo processor, 4 GB of RAM Intel GMA X3100 GPU, 320 GB SATA magnetic drive, and Mac OS 10.7.3 Battery remained in during all experiments.
Dell Vostro desktop: quad-core Intel Core i5 processor, 4 GB of RAM AMD Radeon 6450 GPU, and 250 GB SATA magnetic drive, tested under Windows 7 and Ubuntu 10.04.
Lenovo T410 laptop: Intel Core i5 processor, 4 GB of RAM 300 GB SATA magnetic drive, Windows 7. Battery remained in during all experiments.

Table 4.1: Hardware and software descriptions of the computers used in this work.

To record each workload's power signature, we monitored electrical current between the power supply and an instrumented outlet. To measure a power supply's

instantaneous current on the hot–neutral circuit, we placed a 0.1 Ω *sense resistor* (part #13FR100E-ND) in series with one terminal of a standard US outlet. We attached an Agilent U2356A data acquisition unit (DAQ) to the terminals of the sense resistor. The DAQ samples the voltage across its probes and sends the data via USB to another PC (not the computer being measured). We recorded 16-bit samples at a rate of 250 kHz to capture workload artifacts occurring at up to 125 kHz.

Finally, we developed a Chrome extension to automate the repeated loading of a target webpage. The extension repeatedly: opens a new window, pauses, loads the page, pauses again, and finally closes the window. For webpages that did not require user credentials, the script opened browser windows in a *private browsing* mode to purge the browser environment of confounding cached data. To compare webpage identifiability across browsers, we also used the iMacros extension for Firefox [2] to mimic our Chrome extension. We recorded continuously with the DAQ while running experiments. A script with knowledge of the browser extensions' timings chopped the DAQ's output into separate trace files to be used with our classifier. While the majority of the webpages we profiled show no changes within the measurement period, there are notable exceptions. A number of high-turnover webpages including `cnn.com`, `cnet.com`, and `reddit.com` underwent content changes during our measurements.

4.5 Evaluation

This section summarizes our experimental results over a wide range of conditions. While there are a limitless number of questions to ask about how well a classifier works under different conditions, we have distilled them down to the following six questions regarding causality and intuition:

- How effectively can the SVM classifier differentiate webpages from one another? (Section 4.5.1)

- How does sampling rate affect classification? (Section 4.5.5)
- How robust is the classifier in the presence of content distribution services, encryption, and caching, as well as changes in network location, type, or interface? (Section 4.5.2)
- How is classifier performance affected by changes in operating system or hardware? (Section 4.5.3)
- How does the classifier's performance change when the test traces include background activities? (Section 4.5.4)
- How well does the classifier exclude samples of pages outside the corpus? (Section 4.5.6)

We find that our classifier can differentiate webpages with high precision and recall rates (each averaging 99%) and that it is robust against many of the variations we tested, including the use of a VPN, and changes in the location or network interface. It is not robust against changes of machine or operating system. Where our classifier performs poorly, we find in most cases that increasing the diversity of the training set improves its performance along all metrics. The total number of power traces we tested across all experiments was $13,110$, chronicling over 100 hours of 250 kHz trace recordings.

4.5.1 Page differentiation

Our SVM classifier effectively differentiates among the 51 popular webpages we tested. As a baseline for classifier performance, we varied only the webpage under test and held all other variables constant. These other variables include machine under test, network location, network interface, operating system, and web browser. By varying only the webpage under test, we minimize differences that are not actually the result of variation among webpages.

We gathered all of the data for this experiment twice, using slightly different MacBooks and instrumented outlets in different locations. We took this step to ensure that the results were reproducible. Here we present the results of combining the two data sets into a single corpus. After gathering ∼90 traces for each of the 51 webpages on each of the MacBooks (for a total of ∼180 traces), we used the experimental protocol described in Section 4.3.2 to label each trace. Each SVM labels a trace as either matching or not matching the page for which it was trained. The total size of the corpus for this experiment was $9,240$ traces. We used half of these traces for training and the remaining traces for testing. With ∼90 training examples per label, the SVM classifier achieves an average 99% precision and 99% recall over all webpages in the data set.

The classifier's performance did vary among the tested webpages. The precision and recall were both 100% for 18 of the 51 webpages. The lowest precision for any page was 93% for `skype.com` and the lowest recall for any page was 88% for `slashdot.org`. A plausible explanation for the classifier's relatively poor performance on `skype.com` is that the front page underwent a significant design change between the capture of our two data sets, which we confirmed by inspecting snapshots from the Internet Archive Wayback Machine [37]. The poor recall for `slashdot.org` could be explained by the high turnover of the front page or inconsistent load times. `godaddy.com`, which uses a commercial content distribution service, also yielded lower recall results.

4.5.2 Diverse browsing conditions

We varied the conditions under which our browser operated and found that the SVM classifier is robust against local network connection type, use of cache, VPN encryption, and the passage of time for most webpages. It is not robust against the use of a caching content-distribution network (CDN) such as Coral [27].

Our training and testing setup was as follows. We repeated this process for three webpages: one simple page (`google.com`) and two complex pages (`cnn.com` and `cnet.com`). For each page, we gathered the following sets of traces on one of our MacBooks:

- **Time:** Traces gathered a month later, to test how fresh the training set must be.
- **Cache:** Traces recorded with a warm browser cache, to test whether the classifier depends on specific network traffic patterns.
- **VPN:** Traces recorded while connected to a VPN concentrator a 1.5 ms round trip away (essentially measuring only cryptographic overhead), to test whether encrypting normal traffic would be an effective countermeasure.
- **WiFi:** Traces recorded while connected to our lab network wirelessly instead of via wired Ethernet, to test whether the training phase overfits the SVMs to "clean" low-latency wired traffic.
- **CDN:** Traces recorded with web traffic passing through the Coral CDN, to test whether a caching proxy sufficiently disguises traffic.

To test each one of these sets, we trained an SVM on all of the *other* sets using only samples from the same MacBook. For example, to test whether the SVM could correctly label traces in the *WiFi* set, we trained the SVM on the *Time*, *Cache*, *VPN*, and *CDN* sets in addition to the *Base* set of samples. In contrast to training only on the *Base* set, this training approach avoids overfitting the SVM to that set. After training the classifier, we instructed it to classify the traces in the test set.

The only condition that hurt performance, for two of the three webpages, was the use of the Coral CDN. For `cnn.com`, the classifier incorrectly labeled all traces from the *Coral* set as negatives; for `google.com`, the classifier incorrectly labeled 45 of 50

traces as negatives, resulting in a 10% recall rate. The classifier's poor performance on Coralized pages illustrates that, while the network interface's power consumption may not uniquely determine how a trace is classified, the network interface may still alter the timing of characteristic consumption patterns for *downstream* devices such as the CPU and GPU that act on its outputs. Another variable that is difficult to observe is the effect of repeated page loads on the CDN itself. There is no guarantee that the same node will serve each request and the cache conditions of the node are unknown. Coralizing `cnet.com` likely made little difference in its classifiability because `cnet.com` is already distributed via a commercial CDN.

The classifier's performance on traces from the *VPN* set deserve special attention. They suggest that encryption and decryption, at least as implemented by our MacBook's PPTP VPN, have little effect on power-trace classification—i.e., the SVM classifier is robust against VPN encryption.

We also isolated the effect of changes in network location. The Coral results suggest that changes in latency or throughput alter packet arrival times enough to thwart the classifier. To test this hypothesis, we created two static pages that do not have any running scripts or asynchronous content and gathered traces of one MacBook loading each page in different locations. One location had a residential connection and the other a university research network connection. We then trained on all pages in the corpus, including samples of the static pages using only one of the two locations. We tested on samples from the untrained location.

When trained on the residential connection and tested on the university connection, the classifier's precision and recall were 100% and 73% respectively. This result shows that the training did not lead to any false negatives for other pages, but was not able to identify all samples. When we reversed the training and testing locations, the precision and recall were both 100%. This experiment demonstrates that it is not always necessary to use or simulate a potential victim's connection to train an

effective classifier, but that the network connection's impact is determined largely by page content.

4.5.3 Operating system and machine diversity

Training on traces from a single operating system limits the classifier's effectiveness to that operating system. However, training on traces from two operating systems allows the classifier to correctly identify traces gathered using both OSes. To test this behavior, we gathered traces of `google.com` and `cnn.com` using Windows 7 and Linux (Ubuntu 10.04) on the desktop PC. For both sets of traces, we used the same versions of the Chrome browser and our custom Chrome extension for test automation. When trained only on examples from one operating system, the classifier failed to correctly label traces from the other. The only exception was a single trace of `cnn.com` loaded from Linux, which the classifier identified correctly despite the having been trained only on examples from Windows 7. When we rearranged the input sets so that each contained an equal number of traces from each OS, then trained on one of these mixed sets, the classifier correctly labeled all unlabeled traces from the other mixed set.

Differences among OSes include system timers, drivers, memory management, GUI characteristics, and performance tuning. Versions of the same application built for different platforms may also differ in unspecified ways. All of these differences may play roles in differentiating power-consumption patterns. The above result suggests that a prospective attacker should collect traces under as many different operating systems as possible.

When we varied both machine *and* operating system, the SVM classifier failed to correctly label any traces. We trained an SVM on the MacBook (running Mac OS 10.7.3) with 50 webpages and tested on the Lenovo laptop (running Windows 7) for 5 webpages (`google.com`, `cnn.com`, `espn.com`, `live.com`, and `youtube.com`). Then switched the roles and trained and tested again. For all webpages, the SVM failed to

correctly label many traces from one machine when trained only on examples from the other. The precision and recall never exceeded 10%.

Training on examples from *both* machines allowed the SVM to classify traces from both machines accurately: 98.4% precision and 98.2% recall on average for the 5 webpages. This result suggests that, as in the operating system experiment, the problem lies in the lack of training diversity. In the future, we intend to test this hypothesis by training an SVM on a small, but diverse, set of machines and then testing traces from machines that are not represented in the training set.

4.5.4 Background activities

Noting the tendency of users to browse multiple webpages at the same time and running background processes, we measured the classifier's sensitivity to background noise. We chose one of the 51 webpages in our training set—`live.com` —and loaded it with combinations of background processes. We collected traces for `live.com` on a MacBook when a combination of the following 4 processes were running: `gmail.com`, iTunes radio, `pandora.com`, and a word processor. We collected traces for 8 combinations in total, e.g., only `gmail.com`; `gmail.com`, `pandora.com`, and word processor together, etc. We trained the SVM with examples for all 51 webpages without any background process and tested it using the background examples. In all cases, the classifier was able to classify `live.com` accurately with 100% precision and 100% recall.

Even though we only tested one webpage and a small set of background processes, the result suggests that the classifier can be robust against combinations of background processes. A possible explanation for the classifier's robustness is that the background processes do not saturate the CPU load and have little effect on the GPU load because they are literally in the background and do not draw to the screen. Quantifying the limits of this robustness will require further investigation.

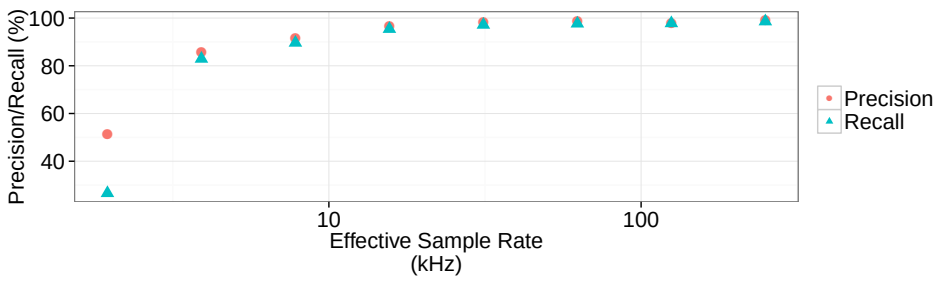


Figure 4.4: The average precision and recall across all 51 pages with exponentially increasing sample rate. The classifier's performance decreases with sampling rate, but the precision and recall do not drop below 90% until the sampling rate is less than 4 kHz, a 60x reduction.

4.5.5 Sampling rate differentiation

Decreasing the sampling rate at which an instrumented outlet records voltage would allow for tracing and exfiltration using simple, low-cost hardware. To understand how robust the classifier is to changes in sampling rate, we repeated the set of page differentiation tests, but simulated lower sampling rates by restricting the set of input features to those representing lower-frequency components. Figure 4.4 compares the results with the original sampling rate against results with simulated lower sampling rates. Each reduction in sampling rate is by a factor of two.

Reducing the sampling rate by a factor of more than 30 (from 250 kHz to 7.8 kHz) incurs only a 9% reduction in average precision and recall. These results show that the lower frequency bands alone contain enough information to accurately classify webpages. This stands in contrast to the work of Enev et al., which specifically targeted changes in power supply switching frequency as an information source [19]. An attacker could likely produce a compact and inexpensive measurement device capable of mounting successful attacks.

4.5.6 Exclusion of unknown pages

Our classifier reliably identifies pages appearing in the training set, but a practical attack would require the classifier to reject pages *not* appearing in the training set

as well. With training and testing sets that resembled each other, a classifier could perform equally well in the previous experiments whether it learned to cluster positive or negative examples. To test the hypothesis that the SVMs learned to cluster only *negative* examples during training, we tested them with a set of previously unseen webpage samples that were not in the training set.

We gathered one trace from each of 441 webpages randomly selected from a list of 1 million popular pages published by Alexa [4], making sure to remove pages that were already in the training set. We then tested all 441 pages against all 51 trained SVMs and measured their false-positive rates. The total false positive rate over all classifiers was 1.6%, leading us to reject the above hypothesis and conclude that the SVMs correctly learned to cluster positive examples.

4.6 Countermeasures to Limit Leakage

This section sketches several countermeasures to mitigate the threats described in Section 4.1.1. Hardware and software countermeasures both present inherent trade-offs. Hardware mechanisms that increase design complexity or cost may not find traction with high-volume manufacturers. Software countermeasures that increase computational work may vitiate energy-efficiency measures. Altering workloads to disguise activity may negatively affect usability or user satisfaction. Effective yet usable countermeasures remain an open problem.

4.6.1 Software countermeasure: Cover activity

To thwart an eavesdropping adversary, a potential victim may be able to hide a sensitive task's power signature by running additional tasks that increase the system's power consumption, which we call *cover activity*. Section 4.5.4 shows that cover activity is not a guaranteed fix. Unless it saturates the available resources by maximizing systemwide power consumption, a cover task might be easy to filter or

subtract from the AC signal. The ElectriSense system offers some evidence that this subtraction is feasible, even if cover activity is sufficient to foil the classifier without modification. Because of the shifting noise floor in typical homes, ElectriSense identifies dominant frequencies and subtracts background noise in the frequency domain to improve classification performance [29]. A random-activity strategy would likely share the constant-activity strategy's drawbacks while also potentially leaving sensitive artifacts unconcealed. A carefully constructed cover task could track the sensitive task's activity and attempt to conceal it only during active periods—akin to jamming in wireless communications—but this targeted approach might still leak information via a timing side channel.

4.6.2 Software countermeasure: Delays and throttling

We adopt a defensive idea from Song et al. [68]: introducing random delays in the time domain, which will cause changes in the frequency domain that may confuse our classifier. The classifier's poor performance on Coralized pages (see Section 4.5.2) suggests that delays complicate classification. The problem with random delays, as Song et al. point out, is that different instances of the same private signal, with different random delays added to each, give an attacker enough information to learn the true timing of the signal by simply averaging the delays. The same problem afflicts the defensive strategy of randomizing the order in which the browser loads page elements. A more effective method of concealing true timing information is to batch activity into discrete buckets that are activated (or sent) at a steady rate no matter what.

4.6.3 Hardware countermeasure: Current filtering

Filtering circuitry that damps current fluctuations could prevent workload-dependent information from leaking onto the AC power line. SMPSes internally implement low-pass filters to remove high-frequency noise and meet government EMI standards. Our

experiments reveal that, for the SMPSes we tested, the frequencies useful for classification are below the internal filter's cutoff. A more aggressive low-pass filter or a high-pass filter could remove additional information, but would likely increase the cost and physical size of an SMPS. Our sampling rate experiments show that the classifier is effective until the maximum observable frequency drops below 4 kHz, so much more filtering would likely be required.

4.7 Related Work

This work focuses on classifying run-time events on the order of seconds on a general-purpose computer, in contrast to previous work that measured on–off transitions at a large granularity from a household vantage point. Unclassified research on recognizing activity by measuring AC power goes back to at least 1989, when Hart proposed *nonintrusive load monitoring* (NILM) to map changes in total household power consumption to appliance activations [32, 31]. Hart also recognized the potential for abuse of NILM techniques. Recently, Gupta et al. proposed ElectriSense, a nonintrusive system that uses a single-point monitor to detect electromagnetic interference (EMI) generated by consumer electronics' switched-mode power supplies [63, 29]. Analyzing frequency-domain power signatures, ElectriSense advanced prior work by detecting nearly simultaneous device activation. Both NILM and ElectriSense effectively capture and identify *on* and *off* events at the device level, but neither aims to infer the internal states of integrated commodity devices such as personal computers, as our work does.

Enev et al. refined the ElectriSense concept by studying the correlation of EMI with video signals being played on modern high-definition televisions [19]. Their classifier isolates the television's power supply switching speed as a feature and classifies signals (video segments) more than 15 minutes long. In comparison, we focus on classifying signals containing shorter periods of activity and monitor comparatively

complex general-purpose hardware. Our sensing apparatus is also somewhat simpler, relying on a single sense resistor and no hardware filtering.

Wang presented the Shazam system, which uses spectral fingerprinting to identify short (∼10 second) audio clips [80]. Shazam extracts fingerprints by selecting local maxima in spectrograms and finds the closest matching fingerprint in a pre-trained database. Like our work, Shazam is robust to typical noise sources. Unlike our work, the source signal (audio clip) on which Shazam operates is explicitly intended to carry information and to sound similar regardless of which device transmits it.

Past work has, like ours, exploited side channels to learn sensitive information from traffic that may be encrypted. From previous work we borrow the intuition that webpages induce characteristic activity patterns that are robust against encryption and the passage of time. Several researchers have trained classifiers on encrypted or obfuscated web traffic and observed that they could match webpages against their training set using only packet-length information [33, 52, 54, 70]. Our classifier uses AC power traces as input rather than network traces, and so observes a noisier, aggregate side channel.

Our classifier performs analysis in the frequency domain, unlike time-domain classifiers that take temporal context into account when trying to identify a component signal. For example, recent work by White et al. on classifying VoIP packet sequences into spoken phonemes demonstrated that a trained classifier with knowledge of language properties can reconstruct textual content from encrypted VoIP streams with surprising accuracy [82].

Our work focuses on leakage via a wired channel, unlike many past works that focus on leakage via parasitic modulation. Looking at CRT monitors, van Eck published the first unclassified side channel analysis work, demonstrating that the screen image could be reconstructed remotely using a TV receiver and hand-tuned oscillators [77]. Kuhn further analyzed leakage from CRT and LCD monitors based on parasitic mod-

ulation [48, 49, 50]. More recently, Vuagnoux and Pasini also investigated leakage via parasitic modulation, though they targeted keyboards rather than monitors and detached their laptop power supplies to avoid interference [78]. Barisani and Bianco independently demonstrated keystroke recovery for PS/2 keyboards by attaching a resistor to the AC power cable, as in our work. They focus only on information from the keyboard and rely on the observation of high-speed switching specified by the PS/2 protocol [7].

Our methods are *not* designed to find key material, unlike past work studying DC circuits that required pin-level access to components or detailed knowledge of the circuits under test. Kocher et al. summarize much of the abundant research on timing and power side channels [43, 44]. The most straightforward of these attacks measures a small portion of the complete system and uses domain knowledge to infer the information being processed. This type of attack requires physical access to the system, knowledge of the cryptosystem under attack, and thousands of accurate measurements of the same process.

4.8 Discussion

4.8.1 Alternative tracing methods

In our experiments, we physically connect probes to an AC circuit to trace electrical activity. An ancillary goal of this work is to demonstrate that it is possible to covertly modify a power outlet, so physical contact with the computer's power cord is a reasonable expectation under our threat model. However, less-invasive methods exist to measure the current along the power cable. In particular, a *Hall effect sensor*, which measures current via the magnetic field around a wire, could provide a way to trace power consumption if modifying the outlet is infeasible. Such an eavesdropper could easily be removed when not in use. We have not tested our classifier against

traces captured with a Hall effect sensor, but we have confirmed that Hall effect sensors matching our sense resistor's sensitivity exist.

Another possibility is indirect measurement similar to that of Enev et al. [19]: connecting measurement equipment in parallel with the victim on the same electrical circuit but on a different outlet. We expect classification performance to decline because of the higher noise floor, but measurements might reveal that traces from outside the victim's outlet are *qualitatively* good enough for an attacker to use.

4.8.2 Adding classification features

The current SVM classifier relies solely on a coarse-grained Fourier transform to learn unique webpage features. There are many promising extensions to the feature space that could improve classification performance. One simple extension would be to increase the resolution of the Fourier transforms used to train and test the classifier. Doing so would increase the dimensionality of the feature space, and possibly the classifier's ability to distinguish among webpages.

An extension that takes advantage of SMPS load characteristics would be to simultaneously sample both voltage and current. As Section 2.2.1 discusses, SMPSes pull the voltage and current waveforms out of phase in a way that is related to the workload. The changing relationship between the voltage and current waveforms over time may reveal more information about the state of the system that is orthogonal to raw current consumption.

Another promising extension is the addition of higher-level features, such as an estimate of how many characters the user typed immediately before navigating to a webpage. While our AC measurements do not reveal user keystrokes in most cases, the address bars of many modern browsers offer an opportunity: when they attempt to do URL completion or even DNS prefetching [47, 18] as a user types, the activity creates distinct spikes of CPU activity after each keystroke. Figure 4.5 illustrates

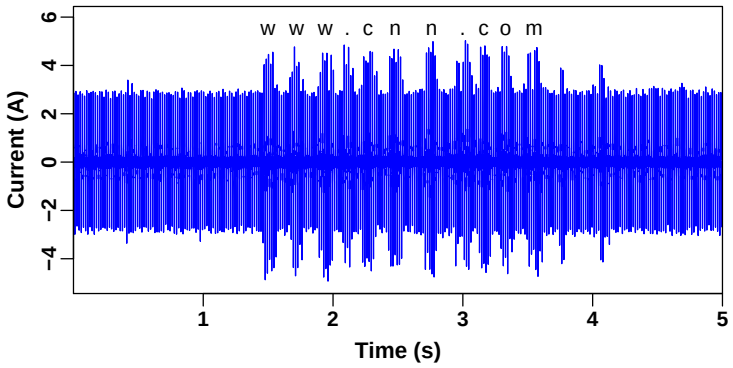


Figure 4.5: Text boxes that trigger CPU activity with each key press yield extra information on the AC power channel. In this trace from Chrome on our MacBook, it is visually evident how many keys we pressed. Even a rough estimate could be used to inform a webpage classifier.

that the spikes are distinct in the time domain. Augmenting the classifier with a URL length estimate could eliminate obviously bad matches.

4.9 Summary

This chapter demonstrates that a computer's AC power consumption reveals sensitive information about computing tasks, specifically the webpage that the computer is loading. We designed methods for webpage identification that extract power consumption signatures that are obscure in the time domain but more apparent in the frequency domain. With a data set of over $13,000$ power traces of 51 popular webpages, our trained classifier can correctly label unseen traces with 99% precision and 99% recall. The power trace signatures are robust against several variations including the use of an encrypting VPN, background processes, changes in network location, or even the use of a different computer.

This is the first work that quantifies the degree to which information about browsing activity leaks via the AC power supply. We believe it represents an early step in understanding this side channel. The increasing dynamic range of hardware power consumption will lead to further information leakage. Open research problems include

the design and evaluation of countermeasures to mitigate the privacy risks of using untrusted power infrastructure.

CHAPTER 5

POWER ANALYSIS FOR MALWARE DETECTION

Power side channels pose privacy risks to many platforms, and there is abundant work demonstrating attacks, as discussed in Chapter 4. Comparatively little work has considered the possibility of *improving* security or privacy with power side channels, proposing its use for malware detection on mobile phones [53, 42], or to detect software theft in embedded systems. [10]. This chapter applies systemwide power analysis to safety-critical embedded systems — presenting PowerTrip, a novel *nonintrusive* malware and anomaly detection system.

PowerTrip addresses the problem of malware on embedded devices that cannot be protected with traditional software- or network-based techniques. This chapter answers the following questions:

- To what classes of devices does systemwide power analysis apply as an anomaly detection mechanism?
- How effective is PowerTrip relative to existing approaches?
- How is performance affected by testing against malware samples for which PowerTrip has not been trained?

We find that PowerTrip applies to a variety of safety-critical embedded devices and, without requiring any software or internal hardware modifications, matches the performance of the state-of-the-art in behavior-based malware detection. When trained and tested on samples of the same malwares, PowerTrip achieves at least 94% accuracy on each of the devices we tested.

5.1 Introduction

Embedded systems perform critical functions in settings from industrial facilities to automobiles to medical devices. An emerging trend is that embedded devices are increasingly connected to other devices—and therefore increasingly exposed to malware [1, 74]. The U.S. Food and Drug Administration has recently acknowledged these risks by issuing a safety communication concerning cybersecurity [75].

Unfortunately, many embedded devices are incompatible with conventional software-based anti-malware mechanisms such as antivirus (AV) programs or networked intrusion-detection systems (NIDS). Traditional embedded devices commonly lack operating systems or network stacks, and they are subject to severe resource constraints that preclude the use of traditional protections. Recently discovered malware infecting supervisory control and data acquisition (SCADA) systems [22] has challenged the assumption that these resource constraints protect against malfunctions.

On the other end of the device spectrum, "embedded" devices based on off-the-shelf OSes—a practice that gives these devices access to filesystems, network stacks, and so forth "for free"—are commonly off limits to their owners because manufacturers will not support third-party software. Some manufacturers even proscribe software updates [9]. The fundamental tension for owners of these devices is that they can have the devices they need to perform critical functions, but they cannot adequately protect the devices using conventional, software-based means.

In light of the above problems, it is desirable for owners to be able to verify that an embedded device is performing *only* its stated duties, and performing them properly—not, for example, joining a botnet or thrashing a subcomponent. Performing this verification *nonintrusively*—without requiring that manufacturers change software or hardware—is the primary challenge.

A key observation is that many embedded devices perform simple, repetitive functions, such as periodically actuating an electrical relay, controlling a pump, or col-

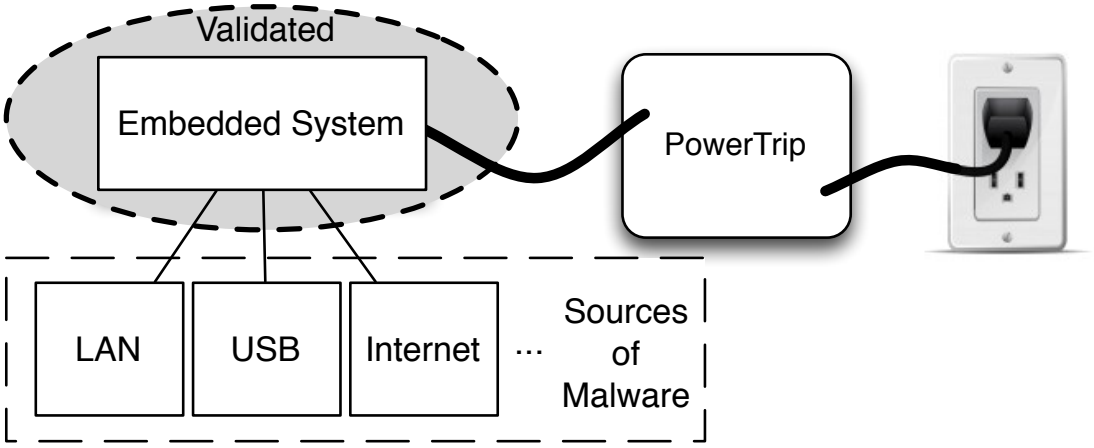


Figure 5.1: Unlike existing malware detection systems, PowerTrip requires no hardware or software modifications to the embedded system under test. Safety-critical embedded systems undergo strict validation processes that make it very difficult to add third party software in practice to anything in the validated space. PowerTrip operates on its own terms outside of the validated space.

lecting sensor readings. Even devices based on off-the-shelf OSes typically perform simple functions implemented as applications on top of the client software rather than in firmware. In both cases, the externally visible state space is small.

This chapter addresses the challenge of malware on embedded systems by introducing *PowerTrip*, a nonintrusive monitoring system for embedded devices. PowerTrip relies on the *side channel* of systemwide power consumption, which leaks information about the system's computing activity. It is well suited to embedded systems because of their typically constrained state spaces, which manifest as a small set of discrete power-consumption levels.

Designers of trustworthy systems typically seek to *eliminate* side channels on principle, because of well-documented risks such as key disclosure or unintentional leakage of private information. Components' power consumption as an *undesirable* side channel is well established [44, 19, 14]. However, in addition to sensitive information, side channels can also leak *constructive* information. Many computing devices exhibit systemwide power consumption that scales closely with their workloads. Systemwide

power consumption, as a proxy for computing activity, can be used to nonintrusively detect problems, including malware. Figure 5.1 gives a high-level overview of how PowerTrip works.

5.1.1 Contribution

This chapter's contribution is PowerTrip, a machine-learning–based system that monitors the behavior of embedded systems by analyzing systemwide power consumption. We test PowerTrip on three types of embedded hardware: a programmable logic controller (PLC); a PC-based SCADA substation computer; and a pharmaceutical compounder, a Windows-based medical device. In each case, we trained PowerTrip's classifiers on both normal and abnormal behavior, and later tested on unlabeled samples of both behaviors. PowerTrip correctly distinguished between the two with greater than 98% accuracy on the PLC and substation computer, and greater than 94% accuracy on the compounder—comparing favorably to the 55% detection rate of a commercial antivirus product and the 86% detection rate of a recent behavior-based malware detector [26], yet without requiring software or hardware changes. For PLCs not based on off-the-shelf operating systems, this work fills a gap by providing malware detection where none existed, addressing the risk of targeted attacks like Stuxnet. For SCADA and medical hardware based on off-the-shelf computers and OSes, PowerTrip provides a malware-detection scheme that is less intrusive than conventional antivirus products.

5.1.2 Lessons Learned

Using PowerTrip as a starting point, this chapter offers the following broader lessons that may apply to other domains:

- Systemwide power consumption, which typically causes privacy anxieties from users, can be constructively used for anomaly detection, especially for systems with dedicated tasks.

- Using supervised learning approaches, our study of PowerTrip shows that power-analysis-based classifiers can be trained on limited types of anomaly power traces, and yet be able to detect new or unknown malware, precluding the requirement of malware signature updates.
- Although PowerTrip is currently a prototype system that operates offline, our empirical results indicate that power traces can be processed quickly enough to be deployed as a "live" malware detector.

5.2 Vulnerable Embedded Systems

Embedded systems play a vital role in many operations, from SCADA deployments that control physical infrastructure to manufacturing floors to hospitals. These systems are commonly "hands off" for customers, i.e., they prohibit customers from changing software configurations, for several reasons. First, manufacturers are unwilling to support software that they did not install and have not validated. Second, customers are disinclined to risk breaking systems that are working, when defending against unclear threats has an unclear benefit. Third, many devices use specialized hardware and software that may not match customers' assumptions.

The key challenge to resolving the tension between functionality and the need to protect increasingly connected devices is finding a way to monitor devices' behavior without modifying them. More specifically, given a device that may contract malware, the challenges are: (1) inferring the device's state without instrumenting or otherwise modifying it; (2) recognizing deviations from these acceptable states, ideally without a database of known deviations; and (3) minimize the operational burden by avoiding misclassifications.

This section describes two examples of embedded systems—SCADA systems and medical devices—and outlines the threats that malware poses to them.

Device	Power	Configuration
Siemens S7 1214C DC/DC/DC PLC	DC	No OS, MCU, several KB ROM
Siemens S7 1214C AC/DC/RLY PLC	AC	No OS, MCU, several KB ROM
Baxa ExactaMix 2400 compounder	AC	WinXP Embedded, Via 664 MHz , 512 MB RAM
Schweitzer SEL3354 substation computer	AC	WinXP Embedded, Athlon 2600+, 2 GB RAM

Table 5.1: Devices against which we tested PowerTrip.

5.2.1 SCADA Systems

PLCs, described in Chapter 3, are ubiquitous in industrial and commercial automation settings, where they read sensors and control actuators. PLC firmware programs are structured as loops, each iteration of which encodes I/O or computation. PLCs traditionally run a single firmware program without a multitasking OS underneath. A typical PLC comprises a power supply, a CPU, and I/O modules that contain solid-state relays, transistors, and digital–analog conversion components. After an operator loads compiled firmware onto the PLC via a temporary wired connection, the firmware program runs as soon as the PLC is powered.

The Stuxnet malware famously targeted PLCs, covertly changing operating instructions on their way from a programming environment to the PLC, apparently with the intent of causing physical damage [22].

A substation computer is a "ruggedized" commodity PC that uses custom protocols to control PLCs via an array of communication ports, while communicating with a larger network via Ethernet or similar. Substation computers typically run "embedded" versions of mainstream OSes.

This chapter focuses on PLCs and substation computers because both perform relatively few tasks at once, simplifying the task of nonintrusive monitoring, yet they

also form the backbone of infrastructure systems. For this work, we tested AC- and DC-powered variants of a common PLC, and a substation computer running Windows XP Embedded.

5.2.2 Medical Devices

The category of medical devices encompasses a variety of clinical and therapeutic systems. Over half of these devices include software [24], and many have proven vulnerable to garden-variety viruses and malware [72]. Many vulnerable medical devices are directly responsible for patient care, including tissue oximeters, fetal monitors, and pharmaceutical compounders [1].

For this work, we tested a pharmaceutical compounder (Figure 5.2) we obtained from an auction. A compounder mixes precise amounts of liquid ingredients according to pharmaceutical formulations. The compounder runs Windows XP Embedded and custom mixing software. It includes a network port for loading formulations from the network in multi-compounder environments. It also includes specialized hardware to support its mixing function, including a pump driven by an electrical motor, pressure sensors, and a scale to ensure that mixtures meet weight expectations.

The manufacturer expressly advises against the installation of software updates or other typical protections [9]. According to a security whitepaper by the manufacturer, owners of the compounder should take *some* precautions, namely a dedicated firewall and subnet per device [9].[1]

The burden of installing a firewall for each device, and the risk of the firewall interfering with device communication, highlight the need for new approaches to security in this domain. Anecdotal evidence from a U.S. Department of Veteran's Affairs (VA)

[1]The instance of the ExactaMix software on the compounder we tested included network paths (for logfiles, etc.) indicating that the compounder had been connected to a Windows network.

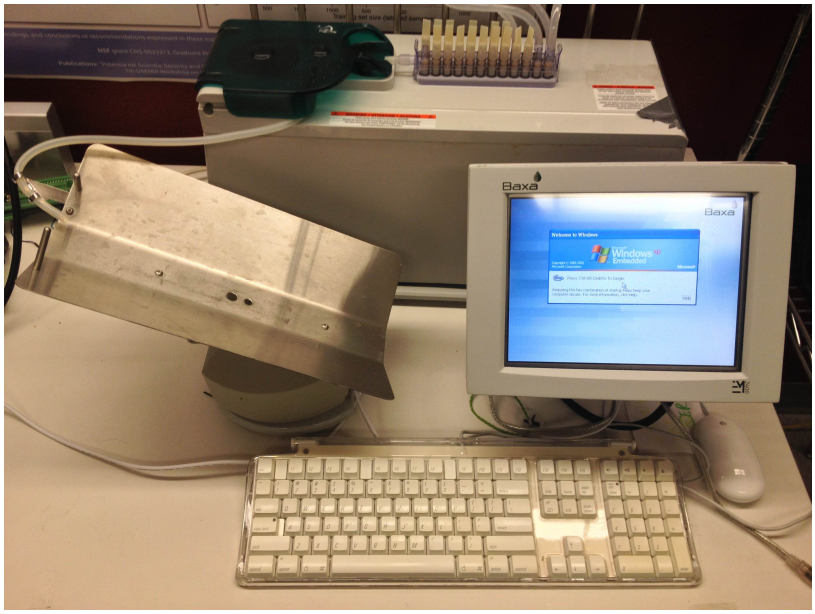


Figure 5.2: Running on Windows XP Embedded SP2, our Baxa ExactaMix 2400 pharmaceutical compounder is an automated embedded system that mixes liquids to individual specifications for intravenous parenteral nutrition.

database suggests that this burden raises the bar too high for IT administrators [67]. The VA database recorded 207 confirmed malware infections over a 35-month period despite IT administrators following the advice of manufacturers to keep devices separate; the network implements thousands of ACLs and uses VLANs extensively.

5.2.3 Threat Model

Because they incorporate both embedded and general-purpose computing devices, both SCADA and medical systems are subject to malware targeting generic off-the-shelf systems *and* to more-specific targeted malware akin to Stuxnet [12, 22].

SCADA or medical systems are exposed to malware if *any* node on the network can accept outside inputs from, e.g., the Internet or a USB memory stick. Computers running off-the-shelf operating systems are of particular concern because they are susceptible to un-targeted as well as targeted malware.

PowerTrip seeks to detect targeted and untargeted attacks, but relies on the assumption that an attacker *cannot* completely replicate both the system under attack and PowerTrip. Such an adversary with detailed knowledge of the defense mechanisms can design an attack specifically to thwart or evade them. Fortunately, these adversaries appear to be rare; we know of no targeted attacks against medical devices in the wild, and only a few examples of targeted SCADA malware have been publicly acknowledged. Using a system like PowerTrip to detect such attacks would require complete knowledge of the system input and outputs, as well as the control flow, to verify state transitions. Cárdenas et al.propose a system with the required device visibility, but their work relies on hardware instrumentation of all system inputs and outputs [11].

Unlike sophisticated targeted attacks, garden-variety malware is a clear and present danger to both medical and SCADA systems [46, 74, 72] and also has the potential to halt industrial operations or interfere with patient care by saturating system resources, causing system restarts, or breaking installed software.

For a PLC, we assume an attacker may obtain the source code to a PLC's firmware, then modify it at any time. An attacker may also change the set of inputs and outputs (read or write extra ports), the values they read or write, and the timings of their outputs.

For a substation computer or a medical device running off-the-shelf software, we assume an attacker may use software exploits to gain administrator-level access.

We assume that devices are initially shipped without malware. We also assume that adversaries do *not* have physical access to the SCADA facility at any time, since the computer systems are perhaps less of a target than the physical systems themselves if an attacker is inside the facility.

5.3 Monitoring Device Behavior with Power Analysis

All of the devices discussed in the previous section share two key properties that make them amenable to nonintrusive monitoring: (1) they perform well-defined, repetitive tasks that should exhibit little variation from run to run; and (2) they draw power from a power outlet. The power outlet can serve as a monitoring point for unmodified hardware. Taking advantage of power outlets as monitoring points, PowerTrip interprets aggregate *systemwide* power consumption as a side channel to determine whether a device stays within the set of states that are correlated with acceptable behavior.

5.3.1 PowerTrip design goals

Previous sections have detailed the motivation for PowerTrip's design. We set the following explicit design goals for PowerTrip:

1. Monitor power *nonintrusively*—do not require device modifications, network connection, or intrude upon their operation.
2. Work well across a variety of systems—AC and DC as well as different kinds of devices.
3. Meet or exceed the malware-detection rate of software-based antivirus products.

To meet the first goal, PowerTrip operates exclusively outside the device being monitored, and takes read-only power measurements of which the device is unaware. To meet the second goal, PowerTrip adopts a machine-learning approach that adapts to each device being measured. Section 5.5 shows that PowerTrip meets the third goal.

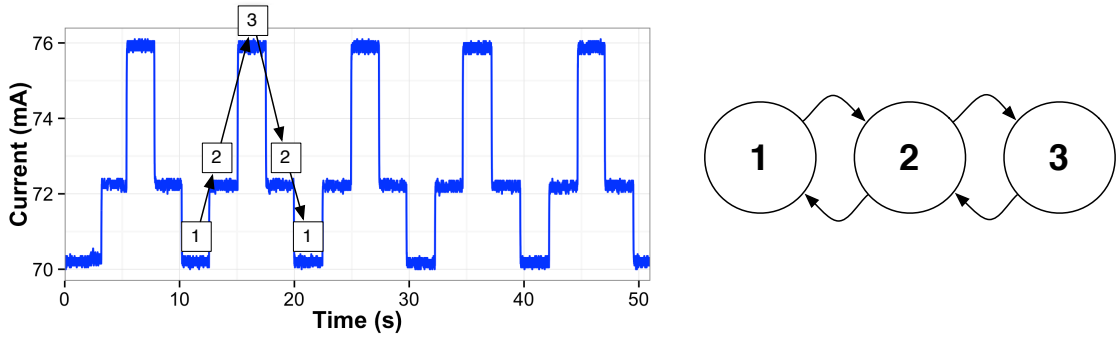


Figure 5.3: A trace of DC power consumption (top) allows inference of a simple state machine for a PLC. Deviations from these states may indicate a malware infection.

5.3.2 Inferring State Machines

Many SCADA and medical devices act as black boxes from an operational perspective, with only their inputs and outputs visible at run time. Accurately determining these devices' states is a matter of inference rather than observation.

An appealing option is to directly learn a device's state machine by observing all of its inputs and outputs during training executions, an approach Cárdenas advocates [11]. While such an approach may work for a given PLC, a key drawback is that it requires custom physical instrumentation for each device and a custom set of classification features corresponding to the particular configuration of inputs and outputs.

An operationally simpler alternative is to monitor power *nonintrusively*, without modifying the device being measured. Because the devices we studied are plugged into a power source, and their power consumption scales with their computing activities, observing systemwide power consumption at the power source gives an *aggregate* view of device activity that maps to the system's constrained state space.

Figure 5.3 depicts the inference of a state machine from a power trace. Assuming that the trace is representative of the device's correct operation, a simple automatic mechanism could infer that (1) correct behavior is periodic (with a correct period

ΔT), and (2) the device's instantaneous power consumption should match one of a few values.

A general-purpose computer is complicated in the sense that it has a tremendous number of possible states. We assume that fully inferring the internal state of a computer by inspecting its power consumption is impractical in the general case. However, control systems that run industrial or medical applications are typically dedicated to a single application per machine, effectively constraining the state space to the application's.

Thanks to this simplification of the state space, we can characterize normal behavior for the industrial-control and medical systems we examine in this work.

- A PLC exhibits periodic behavior with period at most the time between one loop iteration and the next, or it performs a function to transform its inputs into outputs.
- A substation computer periodically collects information from PLCs via wired interfaces, and periodically reports information to higher-level systems.
- A compounder stores chemical recipes (possibly on network shares), mixes chemicals, and verifies the output. It also supports "flushing" to clear its fluid channels of trace chemicals.

Activities other than these may appear in traces of the whole system's power consumption, in which case a classifier can flag them as unusual.

While these state descriptions may seem simplistic, the devices we consider have narrow intended purposes. In the case of the compounder, a single custom application opens full-screen at system launch and is intended to be the only user interface to the device. Outside of variations in the mixed chemicals, there are few user interactions that should affect power consumption.

For a concrete example of normal versus abnormal behavior, Figure 5.4 depicts several power traces of a substation computer running Windows XP Embedded. At idle,

(Figure 5.4(a)), the system's power consumption is constant (modulo some noise). However, after contracting the widespread "Ramnit" virus that attempts to join a botnet (Figure 5.4(b)), its power consumption at idle periodically exceeds the normal range. Under the control of a different piece of malware, the power consumption at idle exhibits more-pronounced periodic behavior (Figure 5.4(c)).

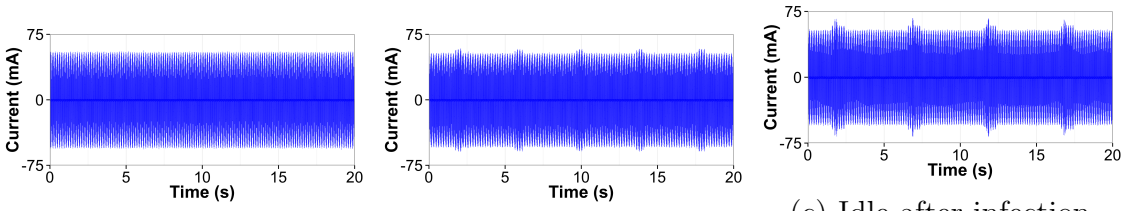
5.3.3 AC versus DC

Both alternating-current (AC) and direct-current (DC) power are in widespread use in SCADA installations, so PowerTrip employs classification strategies for traces of both kinds. The AC classification strategies apply to medical devices, which tend to use AC power throughout because of the wide variety of separately sourced equipment they must support.

A DC power trace is a sequence of levels with step changes between them, and is amenable to relatively simple pattern-matching approaches that map to intuitive concepts. An AC power trace is a 60 Hz periodic signal that undergoes amplitude and phase distortions in response to changes in consumption. Amplitude changes are more prominent than phase changes because they are directly proportional to power consumption. Phase changes are not directly proportional to consumption, but are instead artifacts of power supply switching inefficiency. Because switching and efficiency are both related to consumption levels, phase changes are still correlated with load. AC traces are not amenable to simple pattern-matching because of periodicity and phase misalignments. Because of this, our techniques for AC power traces do not rely on any distance metrics that rely on precise alignment.

5.3.4 Recognizing Deviations

Antivirus scanners identify malware using signature matching to identify unknown code specimens, requiring an exact (rather than heuristic) match to known malware for a specimen to be flagged as suspicious. The major limitation to this approach is



(a) Idle

(b) Idle after infection with "Ramnit" malware

(c) Idle after infection with custom malware that captures a screenshot every 5 seconds

Figure 5.4: AC power traces collected on a substation computer running Windows XP Embedded in a SCADA testbed.

the need to create and distribute signatures for each piece of malware before it becomes widespread. Furthermore, recent work has demonstrated that signature-based antivirus scanners are easily fooled by simple mutations to known malware [38], resulting in false negatives. In the settings we consider, excessive false negatives may have damaging repercussions. For this reason, and also because signature databases require frequent updates from online sources, we seek alternatives to the naïve signature-based approach.

The next section describes a behavior-based approach that offers better detection rates than commercial antivirus software and does not require the device owner to periodically update a blacklist.

5.4 Power-Trace Classification

PowerTrip monitors systemwide DC or AC power consumption at run time. For DC power, PowerTrip uses a whitelisting approach (Section 5.4.1) that requires only traces of normal activity for training. For AC power, which is less amenable to direct time-series analysis, PowerTrip uses a supervised-learning approach (Section 5.4.3) that requires traces of normal and abnormal activity.

5.4.1 Whitelisting for DC Classification

DC power distribution is common in industrial environments, in part because it simplifies machine design (by omitting AC-to-DC transformers). Many PLCs support DC line power for this reason.

Because of its simple structure, a PLC's DC power consumption is strongly correlated with its workload. Additionally, predictable, repetitive, *deterministic* operation is a primary design goal for PLCs, suggesting that a whitelisting approach is appropriate for recognizing the small, predictable set of known-good states.

5.4.2 Classifying traces

A naïve classifier might accept an expected period for repetition and flag any departure from that period. Because PLC workloads may or may not be purely periodic, PowerTrip instead uses a hybrid classification approach. Given a representative training trace, the classifier's first stage searches for periodic content. If it finds periodic content, it chooses a classification strategy that searches for the same periodic content in unlabeled traces. If it fails to find periodic content, it switches to a clustering classifier that determines whether samples in an unknown trace fall into the same range of power levels as training samples. More detail follows.

A common task for PLCs is to control actuators with regularly spaced electric pulses. The frequency of the pulsed output determines the rate of the actuator, with one common output type being a stepper motor.[2] Our classification approach for these purely periodic workloads is to automatically identify one repetition of the signal, then try to match this repetition against future traces, as follows:

1. Set $n = 1$ (begin with a one-sample candidate window).

[2]Stuxnet targeted pulsed outputs, increasing the pulse frequency and then abruptly stopping them in order to damage centrifuges attached to stepper motors [22].

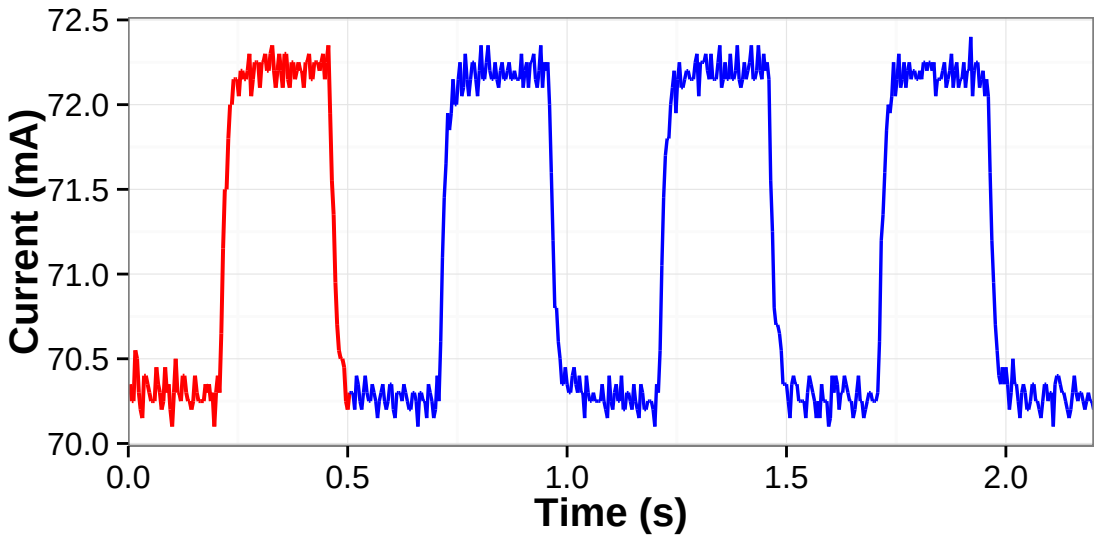


Figure 5.5: PowerTrip iteratively grows a window to characterize and later check pulsed outputs.

2. Compare samples $\{1, \ldots, n\}$ from the input trace with the next n samples ($\{n+1, \ldots, 2n\}$).
3. If the sum of squared differences between the two sets of samples is less than a threshold value T, consider them repetitions of the same signal. If consecutive copies of the n-sample sequence match the rest of the trace, mark n samples as the signal's period. Otherwise, increment n and repeat.

Figure 5.5 illustrates the process.

To test whether an unlabeled trace's workload matches the periodic workload identified in the above way, a simple classifier slides the window of n samples across a downsampled trace; if it matches at some location, and consecutive copies of the n-sample window match the rest of the trace with sufficiently few matching errors, the classifier outputs a binary matching decision. A result of "no match" indicates suspicious behavior.

For workloads that are not purely periodic—e.g., those that are determined by unpredictable inputs—a slightly more complicated approach is appropriate. This

approach relies on the assumption that, even if the *order* of power-consumption levels is unknown, their *amplitudes* will still fall into a fixed number of categories. For accurate training, PowerTrip's approach requires the system under test to enumerate its state space so that all expected states appear in a training trace.

Given a trace for training, PowerTrip uses a k-means clustering algorithm as follows. First, it calculates an upper bound for k, $\hat{k}$ corresponding to the maximum number of power-consumption states. DC power traces do not allow us to discriminate PLC inputs or outputs that use identical hardware (see Section 5.5.3), so the maximum number of possible states is not equal to all combinations of the input and output states. The classifier can instead identify *how many* inputs and *how many* outputs are in a high state. The maximum number of states is then n^2 for a PLC with n inputs and n outputs. In the case of the Siemens S7-1200 PLCs, there are 4 inputs and 4 outputs, and thus 16 visible power states.

Having calculated $\hat{k}$, the classifier sets $k = i$ for each $\hat{k} \geq i \geq 2$, to see whether clustering the states into i clusters minimizes the mean squared distance from the set of points to the set of cluster centroids. PowerTrip sets k to this value of i and infers that the system's state machine has k states. (Finding k automatically is an NP-hard problem, but our iterative method produces the desired k for a PLC's small state space.)

After clustering, the trace comprises k probability distributions, each with its own mean $\hat{x}_i$ and standard deviation σ_i. To classify an unlabeled input trace, which is a sequence of voltage samples, we iterate through the samples. For each sample s, we test whether it is within one standard deviation of one of the k means—that is, whether $\hat{x}_i - \sigma_i \leq s \leq \hat{x}_i + \sigma_i$, for any $i \in \{1, \ldots, k\}$. If so, PowerTrip considers the sample "reasonable." If 99% of the trace's samples are reasonable, the classifier deems the entire trace reasonable. Otherwise, it raises a flag to indicate suspicious

behavior. (The 99% threshold is somewhat tolerant of noise and time spent in state transitions, in our experiments.)

5.4.3 Supervised Learning for AC Classification

AC power traces are less amenable to classification via a whitelisting approach because the switched-mode power supplies used to convert AC input to DC power introduce noise that can confound matching against time-series signatures. In addition to noise, general-purpose computers have more-diverse capabilities and correspondingly larger state spaces than single-purpose embedded devices. For these reasons, PowerTrip uses a *supervised-learning* approach to AC power classification, taking both negative (normal) and positive (abnormal) training samples.

5.4.4 Choice of classifiers

Using implementations from the Weka toolkit [30] and libsvm [13], we tested a variety of supervised-learning algorithms for AC trace classification. PowerTrip uses the three classifiers that worked best in our experiments.

- 3-*nearest neighbors (3-NN):* Given an n-dimensional feature vector (set of numbers summarizing a power trace), assign it the label that is most represented among its 3 nearest neighbors in the feature space (from the training set).
- *Multilayer perceptron:* Train a neural network to recognize properties that characterize the labeled feature vectors in the training set, using backpropagation of errors to minimize overall error; then pass unlabeled feature vectors to the neural network and allow the learned edge weights to determine the output class (normal or abnormal).
- *Random forest:* A "forest" of decision trees trained on randomly perturbed feature vectors produce, in aggregate, a single label for an unseen trace by majority voting.

All three act as binary classifiers that separate normal behavior from abnormal behavior. For each AC-powered device, PowerTrip splits each power trace into 5-second chunks to simulate real-time sampling, then applies stratified 10-fold cross validation for training and testing. In the case of a PLC, abnormal behavior could be any change in firmware. In the case of a substation computer or other higher-layer SCADA device, we focus on malware as the most interesting type of abnormal behavior.

The primary advantage of using general-purpose, well-understood supervised-learning algorithms for AC classification is the versatility of the approach: PowerTrip applies the same choice of features and set of classifiers to a PLC, a substation computer, and a medical device without any changes to suit the particular device under test. This approach also requires negative training samples—in our case, traces of malware-infected devices. In Section 5.5 we demonstrate that training on samples of some malware allowed PowerTrip's AC classifiers to recognize samples of other un-trained malware and aberrant behavior, obviating the need to build an exhaustive malware signature database during training.

5.4.5 Feature selection and training

After manually examining AC power traces with statistical tools, we selected a set of time-domain features meant to reflect the types of variation that can occur in an AC signal. While performing routine tasks, we expect the distribution of a power trace and the mean power consumed to remain relatively constant over short time periods. Any unexpected software workloads will place extra strain on the hardware, which will in turn consume additional power and affect feature values. PowerTrip considers eight time-domain features:

- *mean, variance, skewness, kurtosis:* the 1st, 2nd, 3rd, and 4th moments of the data.

- *Root mean square (RMS):* the square root of the mean squared amplitude;
- The global *minimum* and *maximum* values from all samples in a trace;
- *Interquartile range (IQR):* the difference between the 75th- and 25th-percentile values in the trace; and

PowerTrip also uses frequency-domain features. It transforms AC power traces into the frequency domain using the Fourier transform, producing divisions 250 Hz wide. It uses the energy in the ten lowest divisions as features representing periodic content up to 2.5 kHz. Feature f1 represents the energy observed in the interval between 0 and 250 Hz, f2 covers the interval between 250 Hz and 500 Hz, and so on up to f10. Our initial experiments indicated that adding more frequency-domain features did not significantly increase the accuracy of the classifiers. It did, however, increase training time which scales with the number of features.

None of the features rely on precise trace alignment. In the case of the time-domain features, PowerTrip calculates them over a window at least 1 second wide, much wider than the ∼17 ms period of the AC signal. This prevents noise in a single period from affecting feature values.

5.5 Evaluation

PowerTrip must detect malware with high accuracy, few false positives, and false negatives. Our evaluation examines the effectiveness of PowerTrip against malware either known or unknown during training on safety-critical devices including a PLC, a substation computer, and a compounder. Our results can be summarized as follows:

- When trained on samples of both normal behavior and infections of known malware, PowerTrip correctly identifies *known* malware with 99.5% accuracy for the substation computer and 94% accuracy for the compounder. It correctly identifies *unknown* malware with 84.9% accuracy for the substation computer and 88.5% accuracy for the compounder.

- PowerTrip's classifiers relying on time- and frequency-domain features can detect when behavioral modifications (changing frequencies or the number of inputs or outputs) are made to our test PLC firmwares with 99% accuracy. PowerTrip can also reliably detect when a PLC's firmware is reflashed.
- PowerTrip can be used for real-time classification. The AC and DC classifiers can keep pace with the 250 kHz power-trace sample rate; extracting and classifying features as they arrive.

5.5.1 Metrics

PowerTrip uses binary classifiers that label samples as normal ("negative") or abnormal ("positive"). The following standard performance metrics apply:

Accuracy, $tp + tn/$ # samples, is the fraction of correctly labeled samples.

Precision, $tp/(tp+fp)$, is the fraction of positively labeled samples whose labels are correct. It measures the classifier's resistance to false positives.

Recall, $tp/(tp+fn)$, is the fraction of samples that *should* have been positively labeled that *are* correctly positively labeled. It measures the classifier's resistance to false negatives.

In the above definitions, tp and tn refer to true positives and true negatives, and fp and fn refer to false positives and false negatives. A classifier's precision and recall results provide insight into the *types* of errors the classifier tends to make, rather than counting only the *number* of misclassified samples.

5.5.2 Experimental Setup

Our tracing setup is similar to that of previous work [14, 15] on measuring power consumption at the power line. For both AC and DC power tracing, the goals of our equipment design are simplicity and modularity.

For each DC-powered device we tested, we placed a 3 Ω sense resistor (1% tolerance) in series with the DC power supply. We then used an Agilent U2356A data acquisition unit (DAQ) to sample and log the voltage drop across the resistor at a rate of 250 kHz.

The AC tracing setup is conceptually similar to the DC setup. To capture traces of AC power between a device and the power grid, we instrumented a North American power outlet. We placed a 0.1 Ω resistor (1% tolerance) in series with the hot terminal of the outlet. A future version of this hardware could be encapsulated in either an outlet cover or a small power brick with a pass-through outlet.

Using these AC and DC trace-recording methods, we tested PowerTrip with two Siemens PLCs, a Schweitzer substation computer, and a Baxa compounder.

5.5.2.1 PLCs

We tested two PLC training kits.One PLC was an S7-1214 DC/DC/DC PLC that supported pulse-width modulation and pulse-train digital outputs and was powered by a 24-volt DC supply; the other was an S7-1214 AC/DC/Rly model that supported analog output scaling and was powered by a standard AC supply.[3] As mentioned in Section 5.4, the AC and DC configurations required different classification approaches.

We wrote ladder logic programs created with Siemens Totally Integrated Automation Portal V11 to explore the threat model presented in Section 5.2. Two workloads implemented simple logic gates (AND and OR). Two others read analog inputs, either actuating an output when the input reached a threshold or displaying the input value on an analog output. One modulated a pulse-width modulation (PWM) output, the type of signal used to drive peripherals such as stepper motors, at a fixed frequency. In addition to these workloads, we gathered 20 samples of the PLC's power consumption while uploading new code (one of our testing workloads).

[3]Stuxnet targeted the S7-200, a similar PLC [22].

5.5.2.2 Substation computer

We tested a Schweitzer SEL3354 substation computer [66], both at idle and under a variety of real and simulated malicious workloads. The SEL3354 we tested (circa 2003), was powered via a standard AC outlet, and included an AMD Athlon 64 2600+ processor, 2 GB of RAM, and a Compact Flash card for primary storage. It ran the manufacturer's substation software on Windows XP Embedded. The only hardware difference from an off-the-shelf computer is the addition of 16 RS-232 serial ports intended for use with devices such as PLCs.

We gathered AC power traces of the substation computer under a variety of workloads that represent a range of activity, including idling, rebooting, infected with our emulated malware, and infected with real malware samples.

We were unable to profile the substation computer while it was communicating with PLCs because of a lack of available compatible hardware at the SCADA testbed. However, the Schweitzer software was running at all times.

5.5.2.3 Compounder

To evaluate our technique on medical equipment, we purchased a Baxa ExactaMix 2400 compounder via an auction. The model we tested runs the manufacturer's compounder software on Windows XP Embedded 2002 Service Pack 2. The compounder has a 664 MHz VIA C5 x86 CPU and 496 MB of RAM.

We gathered AC power traces of the compounder running a wide variety of workloads, including idling, booting, shutting down, flushing, compounding 3 mixtures, infected with our emulated malware, and infected with real-world malware samples (Section 5.5.5.1).

5.5.3 Change Detection for DC-Powered PLCs

Inspired by Stuxnet, which manipulated a PLC-controlled motor's switching frequency, we tested whether PowerTrip's sliding-window approach (Section 5.4.1) could

detect changes in the DC PLC's pulse-output frequency, an example of a perfectly periodic workload.

For each of 1, 2, 4, 10, and 25 Hz, we trained PowerTrip to whitelist traces of the PLC outputting pulses at that frequency, then looked for erroneous matches against traces recorded with other frequencies. PowerTrip distinguished the different frequencies with 100% accuracy for 10-minute samples of each.

To determine the smallest frequency changes the sliding-window approach could detect, we trained PowerTrip on a trace of 25 Hz pulsed output, then tested traces of *slightly* lower frequencies approaching 25 Hz. When the modified frequency reached 24.94 Hz, representing a difference in pulse-width period of 100 μs, the classifier detected the frequency shift (i.e., decided the traces were different) after 90 pulses—representing a phase drift of 9 ms. Since a small change in frequency will compound matching errors over time, PowerTrip should be able to detect arbitrarily small changes in pulse frequency given at least 9 ms of accumulated drift, and changes resulting in $\geq$ 9 ms of phase drift immediately. The maximum frequency supported by the Siements S7-1200 PLCs is 1 MHz, so maximum number of repetitions possible before detection is $\sim$9000, but the malicious workload would only run for 9 ms total.

To measure PowerTrip's ability to detect changes in a PLC's I/O behavior (e.g., malicious selection of the wrong output for a signal), we applied the k-means clustering approach to the PLC workloads. When trained on any one firmware, the classifier could detect changes in the *number* of inputs with 100% accuracy, but could not detect changes in *which* input and output ports were read or actuated. This is unsurprising given that individual inputs and outputs in fact contain identical relay hardware. In an industrial control setting, incorrect input or output actuation may be obvious based on the attached outputs (for example, an HVAC unit could turn on the heat instead of the air conditioning), but we do not make any assumptions about attached devices in order to keep our approach general.

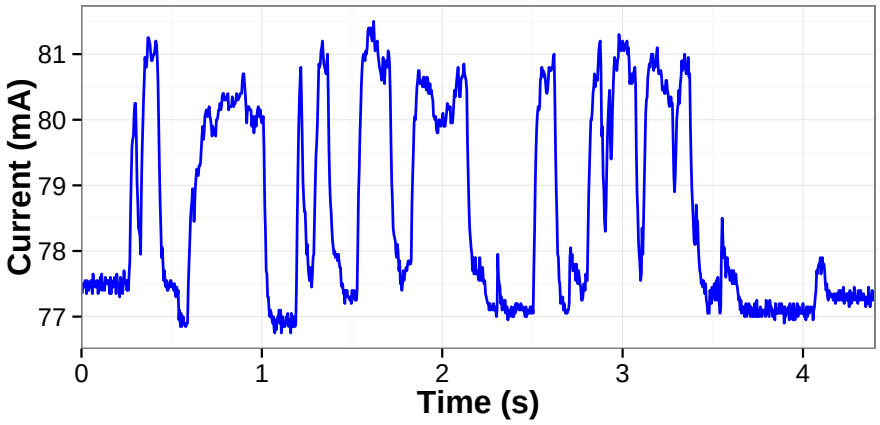


Figure 5.6: During firmware flash operations, PLC power consumption fluctuates much more quickly than it does for non-periodic workloads and much more erratically than it does for pulse outputs.

We also tested 20 traces of PLC reprogramming against all other PLC workloads that were not reprogramming; PowerTrip's k-means classifier for nonperiodic workloads correctly identified all 20 updates as anomalous. Firmware flash traces look markedly different even to the naked eye, as Figure 5.6 illustrates. This experiment suggests that the k-means classifier has a high probability of detecting suspicious firmware updates in real time, even in cases where the new firmware behaves identically to the old.

5.5.4 Extension to AC-Powered PLCs

We can also apply the histogram classifier to AC-powered PLCs given the assumption that state changes occur at less than 60 Hz. Because we have a detailed model of DC PLC traces as a series of voltage levels over time, we can transform AC traces into this form. To evaluate this approach, we took the RMS over a sliding window 4200 samples wide. This effectively low-pass filters the trace, removing changes that occur at greater than 60 Hz. After this process, the traces approximate DC power traces. We tested these pre-processed traces using the k-means clustering approach

and found that it could detect changes in the number of inputs and outputs with 95.6% accuracy—nearly as accurately as it worked for the DC-powered PLC.

5.5.5 Detecting Known Malware

PowerTrip identifies known malware—malware it was trained to recognize—with 98% mean accuracy across all devices tested.

For training and testing of AC traces, we applied stratified 10-fold cross validation. Stratification ensures that each fold in the cross validation process contains approximately the same number of normal and abnormal samples, which is important for repeatable results on biased datasets such as those presented in this chapter.

To evaluate which features best separate samples from each class, we performed feature ranking on the full data set using the Weka information gain attribute evaluator. The evaluator ranks features based on the mutual information between each feature and the class label. The mutual information can be calculated as the difference between the entropy of the feature and the entropy of the feature conditioned on the class label, expressed as: $I(feature; class) = H(feature) - H(feature|class)$ where H is the entropy and I the mutual information. Note that this ranking is feature set dependent, but not classifier dependent.

5.5.5.1 Malware selection

Two types of malware threaten the kinds of embedded devices this chapter considers: *targeted* malware customized for a specific device type that aims to inflict damage on specific targets (but subject to the constraints enumerated in Section 5.2.3), and *nontargeted* malware that finds its way onto off-the-shelf systems via familiar vectors (e.g., USB drives).

To test whether our classifiers could detect deviations caused by nontargeted malware on the substation computer and compounder, we manually installed twelve of the most-prevalent malware programs of 2011 and 2012 [71, 57], starting with the

most prevalent and selecting alternatives when necessary based on sample availability or unsuccessful infection. Each trial began with a "clean" snapshot of the original system. Table 5.4 lists the samples.

To test detection of common malware archetypes, we also wrote emulated malware designed to mimic routine misbehavior. Specifically, we implemented a keylogger, a pop-up dialog launcher that opens dialog boxes (to emulate adware), and a screen grabber that saves screenshots at a fixed interval (to emulate common exfiltration techniques).

For the AC PLC, PowerTrip should flag any change to firmware as abnormal, and thus potentially malicious.

5.5.5.2 Detecting AC PLC Software Changes

To test how PowerTrip's machine-learning methods apply to simple computers like AC PLCs, we constructed a binary decision problem where PowerTrip was trained to identify one firmware versus all others. The firmware chosen at random to act as the "normal" case used an analog output to mirror the voltage level on an analog input. Our AC PLC dataset includes 363 negative samples and 1768 positive samples. Table 5.2 summarizes the results. All classifiers achieved greater than 99% accuracy.

For the PLC dataset, the top five features in order were f6, f8, RMS, mean, and f5, where fn is defined as the energy in the 250 Hz upper-bounded by $n \times 250\,\text{Hz}$ in the frequency domain. These results show that frequency-domain features provides much of PowerTrip's power to differentiate firmwares. The switching speed of an AC power supply scales with power draw, so the relative utility of these features in conjunction with the RMS and mean suggest that changes in overall power consumption could be the dominant identifier in AC PLC traces. Manual inspection of the traces reveals that the PLC's integrated SMPS creates prominent noise in the 1.5–2 kHz range during state transitions. Energy in this range is represented by the two highest-ranked

Classifier	Accuracy	Precision	Recall
PLC			
3-Nearest Neighbors	100%	100%	100%
Multilayer Perceptron	99.9%	100%	100%
Random Forests	99.9%	100%	100%
Substation			
3-Nearest Neighbors	99.5%	100%	100%
Multilayer Perceptron	99.5%	100%	100%
Random Forests	99.5%	100%	100%
Compounder			
3-Nearest Neighbors	93.6%	89%	78%
Multilayer Perceptron	94.4%	99%	73%
Random Forests	94.2%	93%	77%

Table 5.2: PowerTrip distinguishes among our test workloads: accuracy, precision, and recall for our AC devices when we trained PowerTrip on samples of *all* of our test workloads.

features, f6 and f8. From this, we can conclude that changes in power consumption produce obvious switching frequency changes in PLC power traces.

5.5.5.3 Detecting Substation Aberrations

We gathered samples of the SEL3354 substation computer while running the Schweitzer software but otherwise idling, rebooting, and running a wide variety of emulated and authentic malware workloads. The classification task was to separate normal behavior (idle or reboot traces) from abnormal behavior (the malware workloads). Once again, we used stratified 10-fold cross validation for training and testing. There were 2634 total samples tested, 364 negative samples and 2270 positive (infected) samples. All three of the classifiers performed nearly perfectly, with 99.5% accuracy and precision and recall both rounding to 100%.

For this data set, we found that the top five features in order were f3, max, f4, RMS, and mean. Like the PLC dataset, the highest-ranked features all appear to

be related to the overall power consumption or frequency components. The specific frequency-domain features that are ranked highest for the substation computer are different from those ranked highest for the PLC. Given the major hardware and software differences between a largely off-the-shelf platform like the substation computer and a specialized platform like the PLC, differences in the dominant frequency components are unsurprising.

5.5.5.4 Detecting Compounder Aberrations

We gathered traces of the ExactaMix compounder while performing a variety of real-world tasks, booting and shutting down, and while running a wide variety of emulated and authentic malware workloads. We were able to profile the compounder while it performed specialized actions including mixing ingredients according to programmed recipes and flushing the fluid inlets. The classification task was to separate normal behavior (defined as idle, booting, shutdown, or compounding tasks) from the malware workloads. Once again, we used stratified 10-fold cross validation. There were 2343 total samples tested, 1845 positive (infected) samples and 497 negative samples. We used many more positive samples than negative samples because gathering negative samples required manual interaction with the compounder and frequent solution refills. In a clinical deployment, regular use would provide a ready source of negative samples. All three of the classifiers achieved at least 93% accuracy. Table 5.2 summarizes the full results. This experiment demonstrates that PowerTrip's classifiers can reliably separate different workloads given training samples of *all* workloads expected at testing time.

For this dataset, we found that the top five features in order were mean, RMS, skewness, variance, and max. Unlike the other datasets, none of the frequency-domain features appear in the list of top features. Rather, all of the most valuable features for this dataset pertain to the average power draw and slow-moving changes in that

Device	Accuracy	Precision	Recall
Substation comp.	84.9%	98.3%	80.8%
Compounder	88.5%	93.5%	92.1%

Table 5.3: Mean accuracy, precision, and recall over 10 runs for the substation and compounder datasets with malware samples randomly partitioned into two disjoint sets. One set was used for training and the other for testing.

power draw. The dynamic range of the mixing and flushing traces suggest that the compounder's pump is responsible for the relative utility of these features. When the pump switches on, the compounder's total power consumption increases by more than 30%.

5.5.6 Detecting Unknown Malware

PowerTrip identified unknown malware—unlabeled samples of a malware infection that were not in the training set—with 84.9% accuracy on the substation computer and 88.5% on the compounder, which is comparable to the 86% detection rate of a state-of-the-art behavior-based malware detector [26].

We re-trained PowerTrip on half of the compounder and substation datasets after removing all samples of half of the real malware variants. We then tested PowerTrip using only normal samples and those drawn from the malware variants *not* used at training time. That is, the malware variants used at training and testing time were randomly assigned disjoint sets. We partitioned, trained, and tested ten times to avoid a single biased experiment. Finally, we implemented majority voting among PowerTrip's classifiers to produce a single label for each sample. This experiment presents a more realistic deployment scenario, one in which PowerTrip has access to some malware samples at training time but has not been trained on all of the malicious workloads seen at testing time. Table 5.3 summarizes the results, which show that the accuracy decreases by approximately 5% for the compounder and 15% for the

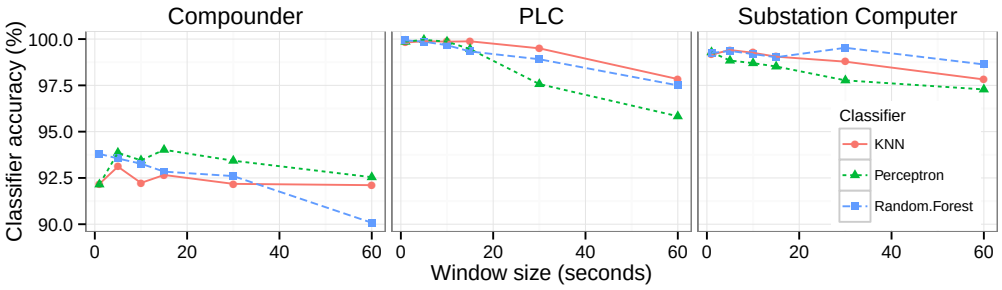


Figure 5.7: The effect on accuracy as the size of the window over which features are calculated increases. Minimizing window size mitigates averaging effects that could hide malware. 5-second windows produce the highest accuracy for all three datasets.

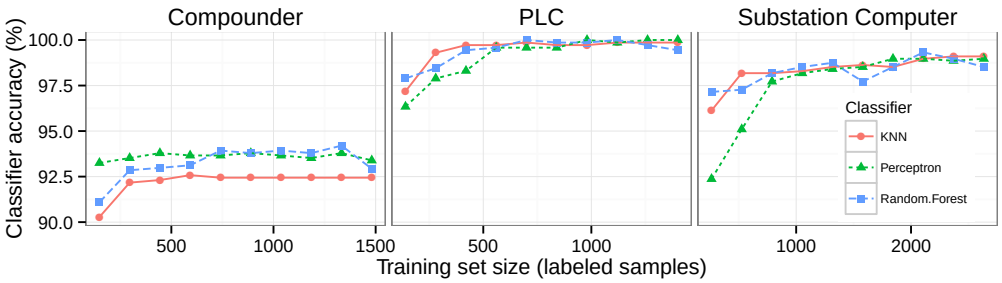


Figure 5.8: The effect on accuracy as the number of training samples increases. For the compounder and substation datasets, real malware samples were included in training data. The point of diminishing returns appears to be approximately 500 training samples for the PLC and compounder datasets and 1000 for the substation dataset.

substation computer, but is still comparable to the state-of-the-art in behavior-based malware detection. The precision is greater than 93% in all cases, meaning that PowerTrip produced few false positives, which could lead to unnecessary downtime in an industrial or clinical setting.

5.5.7 Classification Accuracy

To systematically select the size of the window over which PowerTrip calculates features, we tested six values ranging from 1 second to 60 seconds. Each window size was tested once using the full dataset for each device. The window size presents a

tradeoff in terms of accuracy. Excessively narrow windows could fail to accurately capture longer-running operations on devices like the compounder and excessively wide windows could lower the effective signal strength of short-lived operations, allowing malware to go undetected. For all three datasets, we found that 5-second windows produced near-maximal accuracy for all three classifiers and that the accuracy tended to decrease slowly as the window size increased. Figure 5.7 summarizes the results.

To determine how many training samples PowerTrip requires to reliably separate normal from abnormal behavior, we randomly selected one third of all samples from each dataset for testing and trained on 10–100% of the remaining samples in 10% increments. Each training set size was tested once per device and the training sets contained samples of both normal and abnormal operation. Figure 5.8 summarizes the results, which show diminishing returns in accuracy for the compounder and PLC datasets after approximately 500 samples and for the substation computer after 1000 samples. Assuming 5-second training samples, gathering a sufficient training set would be a matter of only one or two hours assuming that the state space is explored rapidly. In practice, a longer training deployment would probably be necessary to capture all of the acceptable variation.

5.6 Related Work

Cárdenas et al. provide an overview of the challenges endemic to SCADA systems, including the difficulty of modifying them once they are deployed, and propose an approach to SCADA-device anomaly detection that uses knowledge of the system being monitored to develop a template of normal behavior [11]. Unlike our work, Cárdenas et al. evaluate their system in simulation and rely on directly observing the physical devices attached to a control system as inputs and outputs.

Similar to our approach, Cheung et al. advocate model-based malware-detection techniques for Modbus TCP networks and develop an IDS based on this model [76]. Beside our emphasis on power traces instead of network traces, we do not make any assumptions about the protocols in use in the SCADA systems that we seek to protect.

In the area of PLC malware, McLaughlin and McDaniel designed SABOT, a system that generates PLC exploits by matching physical controllers to an adversary-provided model of a control system [55]. Testing our detection techniques against their exploits would be a good next step. They also introduce a method of PLC program analysis that could enhance our techniques' models of normal behavior.

LeMay and Gunter explore the problem of attestation for SCADA systems and propose *cumulative attestation kernels*, essentially bootloaders for embedded systems that provide cryptographic guarantees of software identity [51], akin to Trusted Platform Modules (TPMs) found on mainstream PC hardware. This technique could be useful for assertions of software validity in embedded systems such as PLCs; however, incorporating it in devices would require design changes that may be unpalatable to manufacturers. However, we remain hopeful that an attestation-based approach will enable manufacturers to streamline device upgrades for devices they do not want users to modify.

Mohan et al. describe a system that flags variations of execution time or activation frequency in PLCs and acts to preserve developer-supplied time invariants [59]. Our work uses comparatively complex machine-learning techniques that learn variations on correct behavior. Another key difference is that our mechanisms do not intervene when they detect anomalous behavior; rather, they are meant to provide useful information to administrators even when the systems are not modifiable.

Khan et al. use power tracing to diagnose problems with sensor nodes deployed in the field [40]. They train a Hidden Markov Model classifier on time-domain power traces to identify normal behavior and a variety of failure modes. Our work builds

atop their techniques, but it focuses on a wider variety of devices and must consider adversarial scenarios in which illegitimate patterns may hide in legitimate ones.

Cui et al. propose the use of *software symbiotes*, pieces of integrity-checking code interwoven in embedded software, to detect and prevent malware infections of embedded devices (chiefly routers) [17]. They also propose networking their software into a kind of distributed intrusion-detection grid [16]. These techniques may increase software robustness when policy permits device modification, but we were unable to test their claims because they did not make the source code available.

5.6.1 AC power event recognition

Previous work measured on–off transitions at a large granularity from a household vantage point. We borrow the goal of nonintrusive monitoring from Hart's *nonintrusive load monitoring* (NILM), which maps changes in total household power consumption to appliance activations [31]. In the same vein, Gupta et al. propose ElectriSense, a nonintrusive system that uses a single-point monitor to detect electromagnetic interference (EMI) generated by power supplies [29]. By analyzing frequency-domain power signatures, ElectriSense detects simultaneous device activation, a limitation of previous approaches. Both NILM and ElectriSense effectively capture and identify *on* and *off* events at the device level, but neither targets malware classification.

5.6.2 Activity classification from AC traces

Our work is conceptually related to previous work analyzing power traces from embedded systems. In particular, Enev et al. refined the ElectriSense concept by studying the correlation of EMI with video signals being played on modern high-definition televisions, achieving over 96% classification accuracy with six out of eight HDTV models [19]. They classified signals (video segments) more than 15 minutes long. In comparison, we focus on classifying signals containing shorter periods of

activity. Our sensing apparatus is also simpler, relying on a single sense resistor and no hardware filtering.

5.6.3 DC power analysis

Our methods are *not* designed to find key material, unlike past work studying DC circuits that required pin-level access to components or detailed knowledge of the circuits under test. Kocher et al. summarize much of the abundant research on timing and power side channels [43, 44].

Kim et al. and Liu et al. propose and evaluate power analysis as a malware detection mechanism for mobile phones [53, 42]. Liu et al. present VirusMeter, which uses explicit modeling of a small number of user actions to build a state machine and flag later anomalous power consumption using a variety of classifiers. Kim et al. apply a similar approach that requires offline physical measurements of the device under test to create a suitable model for later use. Kim et al. build signatures for specific pieces of malware and perform classification using the χ^2 distance metric. Like VirusMeter, their system relies on power consumption information provided by the phone that they seek to protect. Our techniques do not require software modifications to the device under test and do not use potentially untrustworthy software APIs to obtain power consumption information.

Finally, this work conceptually extends the recent explorations of AC power analysis by Clark et al. that classified power signatures of webpages with a frequency-domain classifier and proposed applying power analysis to the problem of malware detection [15, 14]. Their analysis focused on general-purpose commodity computing devices and did not incorporate any modeling; we extend this technique to embedded devices that exhibit smaller state spaces.

5.7 Discussion and Extensions

While the power-analysis techniques in this chapter show some promise for revealing malicious activity on embedded or hard-to-change devices, they are not a complete solution to the problem of malware on these devices. This section discusses some of the issues still surrounding real-world deployment of PowerTrip and the generalizability of our technique.

5.7.1 Deployment Scenarios

Because our measurement mechanisms are low cost and nonintrusive, we envision pairing embedded devices with measurement points. As mentioned in Section 5.5.2, an AC measurement point could be a simple "wall wart" form-factor brick with one or more pass-through outlets into which devices could be plugged, or even an in-wall outlet box that looks exactly like a standard outlet plate. In a networked environment, measurement points would stream their readings to a centralized computer for classification and logging. To save traffic, each measurement point could use a low-cost microcontroller to perform feature extraction, then send only sets of feature vectors. Instrumenting only select devices, e.g., those that owners know run outdated software or require network connections, is another possibility that would reduce the management burden while providing increased visibility.

The best strategy for balancing the risks of false positives and false negatives in a clinical or industrial setting is not yet clear. It is a waste of resources for device maintainers to follow up on excessive false positives, but decreasing system sensitivity too far runs the risk of ignoring valid alarms. By designing PowerTrip primarily as an offline reporting tool intended to gather evidence of suspicious activity, we hope to minimize the risk of introducing complexity without providing valuable new evidence of infections.

5.7.2 NIDS

Networked intrusion detection systems (NIDS) are an important monitoring tool for many sites. Unfortunately, most NIDS' detection strategies depend on a certain minimum level of visibility into the systems being monitored, such as access to log entries or the filesystem. As described in the introduction, some vendors prohibit any modifications at all to their systems—leaving NIDS unable to monitor their behavior. We intend PowerTrip to be complementary with NIDS; in fact, it could feed its detection results into NIDS monitoring to provide visibility into systems that otherwise would be unmonitored.

5.7.3 Generalizability

The substation and compounder systems we evaluated run "embedded" versions of the Windows operating system. The key difference versus consumer-grade workstations is that the medical and SCADA systems are designed for specific applications that constrain the computer's state space—and consequently also constrain its power consumption. In contrast, off-the-shelf PCs lack software or policy restrictions to prevent them from running many tasks simultaneously or executing new code. Without a consistent base set of known-good behaviors on a PC, PowerTrip would likely raise false alarms because of an inconsistent or inaccurate internal model. Furthermore, devices without software restrictions can make use of traditional solutions such as antivirus software.

Although it may not generalize to commodity PCs, our PLC results suggest that PowerTrip would likely work well on other embedded devices that rely on firmwares instead of true operating systems. With little or no concurrency, and with state spaces that resemble finite state machines, many embedded devices are comparatively simple to characterize.

5.7.4 Opportunities for Static Analysis

For some single-purpose devices (i.e., those that run a single program rather than a multitasking OS), the program structure may be simple enough to statically analyze to build a model of expected power consumption. Such a model can then inform classification of traces as normal or abnormal. McLaughlin and McDaniel take roughly this approach to build models of PLC operation that help them infer operational maps of SCADA networks [55]. Our techniques are intended to operate on power traces, treating the systems under test as black boxes. With further insight into the firmware state space, however, it would be possible to create a model of the state space or transitions without online training. Given a ground-truth model, PowerTrip could avoid manual selection of parameters or non-deterministic training (as in Section 5.4.1).

5.8 Summary

Safety-critical embedded systems such as medical devices and process control systems need better protection against malware. Protecting these systems without modifying their strictly validated embedded software is a key challenge. This chapter introduces PowerTrip, the first malware-detection system for embedded devices that monitors *systemwide* power consumption. Designed to detect aberrant behavior on devices that exhibit constrained state spaces, PowerTrip uses machine learning to model permissible behavior and detect deviations.

PowerTrip requires no hardware or software modifications to the devices it monitors. We tested PowerTrip on two types of industrial control (SCADA) systems and a pharmaceutical compounder. For known malware, PowerTrip detects malware with accuracy of 100%, 99.5%, and 94.6% on a programmable logic controller (PLC), substation computer, and medical compounder respectively. The extremely high accuracy for the PLC is due to its use of simple state machines implemented directly in firmware, which prove trivially classifiable. For unknown malware on AC-powered

devices, PowerTrip's accuracy degrades to 84.9% and 88.5% for the substation computer and compounder respectively. This performance is still competitive with the state-of-the-art in behavior malware detection. The intuition behind the high accuracy and low false positive/negative rates is that safety-critical embedded systems are designed to do one thing well and repeatedly—unlike general purpose computers. Malware causes subtle changes to power consumption that machine learning techniques can then identify.

5.9 Malware Samples Used to Test PowerTrip

Malware	Activity	Devices
almanahe	Download and execute files	Compounder
autorun	Open port and IE instances	Compounder
bamital	Disable system programs	Substation
bredolab	Download other malware, crash system, unhook API	Compounder
delf	Allow remote access	Compounder
dorkbot	Download other malware, Initiate DOS attacks, Steal personal info	Compounder
fakeav	Download other malware, Ransomware	Compounder
kolab	Spam IRC channels	Both
mabezat	Infect host files	Substation
ramnit	Run malicious routines	Substation
sality	Infect and delete host files	Both
sillyfdc	Disable services	Both
virut	Infect host files, create backdoor	Both
zbot	Steal personal info, create bot	Both
cryptic	Download other malware	Compounder

Table 5.4: Windows malware representing unusual device behavior.

CHAPTER 6

CONCLUSION

Motivated by the long-standing trend toward increasing energy efficiency, and thus energy proportionality, we identify and evaluate security and privacy applications for the *system-level* power side channel. We demonstrate that analysis at this granularity allows detailed inferences about system behavior without software instrumentation or hardware modifications. As energy proportionality continues to improve, this side channel promises to provide richer, more detailed information in the future.

This thesis described an attack against user privacy that leverages standard machine learning techniques to identify which webpage a user is browsing from a trained corpus with greater than 99% accuracy. The attack is effective in the face of many changes in measurement conditions, including the use of a VPN, caching, and changes in the network location and interface. We also identified under what conditions the attack is ineffective, including changing hardware and operating system or browser configurations.

Exploring the possibility of new defensive techniques based on system-level power analysis, we presented PowerTrip, a nonintrusive anomaly detection system for safety-critical embedded systems. After a training period, PowerTrip's machine learning classifiers can identify known malware or firmware reprogramming on the devices we tested with at least 94% accuracy. PowerTripcan also identify unknown malware on the Windows-based SCADA and medical devices we considered with at least 84% accuracy.

We also examined the underlying causes of differentiability in power traces. For relatively simple embedded systems like programmable logic controllers, we found that I/O transitions produce easily observed changes in power consumption amenable to modeling. For commodity computers, we simultaneously measured the AC and DC power channels to identify which power supply outputs most closely correspond to the system-level power consumption and which components these outputs map to. We found that the GPU and RAM are both poor sources of information, but the CPU and motherboard components are more valuable sources.

BIBLIOGRAPHY

[1] MAUDE Adverse Event Report. http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/Detail.CFM?MDRFOI__ID=1621627, Loaded Nov. 2012.

[2] iMacros for Firefox. http://www.iopus.com/imacros/firefox/, Loaded Sept. 2011.

[3] Agilent Technologies. *Agilent USB Modular Products*, July 2010.

[4] Alexa Internet, Inc. Top 1,000,000 sites (updated daily). http://s3.amazonaws.com/alexa-static/top-1m.csv.zip, Loaded Feb. 12, 2012.

[5] Alexa Internet, Inc. Alexa top 500 global sites. http://alexa.com/topsites/global, Loaded Sept. 2011.

[6] Apple Inc. Mac mini — the most energy-efficient desktop computer. http://www.apple.com/macmini/environment.html, Loaded May 1, 2012.

[7] Barisani, Andrea, and Bianco, Daniele. Sniffing keystrokes with lasers/voltmeters. CanSecWest, Mar. 2009. Presentation slides.

[8] Barroso, Luiz André, and Hölzle, Urs. The case for energy-proportional computing. *Computer 40* (Dec. 2007), 33–37.

[9] Baxa Corporation. Preventing cyber attacks. https://btsp.baxa.com/Sales%20Portal/ExactaMix/Preventing%20Cyber%20Attacks.pdf, Loaded Oct. 2012.

[10] Becker, Georg T., Strobel, Daehyun, Paar, Christof, and Burleson, Wayne. Detecting software theft in embedded systems: A side-channel approach. *IEEE Transactions on Information Forensics and Security 7*, 4 (Aug. 2012).

[11] Cárdenas, Alvaro A., Amin, Saurabh, Lin, Zong-Syun, Huang, Yu-Lun, Huang, Chi-Yen, and Sastry, Shankar. Attacks against process control systems: Risk assessment, detection, and response. In *Proc. ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (Mar. 2011).

[12] Cárdenas, Alvaro A., Amin, Saurabh, and Sastry, Shankar. Research challenges for the security of control systems. In *Proc. USENIX Workshop on Hot Topics in Security (HotSec)* (July 2008).

[13] Chang, Chih-Chung, and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 3 (Apr. 2011).

[14] Clark, Shane S., Ransford, Benjamin, and Fu, Kevin. Potentia est scientia: Security and privacy implications of energy-proportional computing. In *Proc. USENIX Workshop on Hot Topics in Security (HotSec)* (Aug. 2012).

[15] Clark, Shane S., Ransford, Benjamin, Sorber, Jacob, Xu, Wenyuan, Learned-Miller, Erik, and Fu, Kevin. Current events: Identifying webpages by tapping the electrical outlet. Tech. Rep. UM-CS-2011-030, Dept. of Computer Science, UMass Amherst, July 2012.

[16] Cui, Ang, Kataria, Jatin, and Stolfo, Salvatore J. From prey to hunter: Transforming legacy embedded devices into exploitation sensor grids. In *Proc. Annual Computer Security Applications Conference (ACSAC)* (Dec. 2011).

[17] Cui, Ang, and Stolfo, Salvatore J. Defending embedded systems with software symbiotes. In *Recent Advances in Intrusion Detection (RAID)* (Sept. 2011).

[18] Dagon, David. DNS security: Lessons learned and the road ahead. Invited talk, USENIX Security Symposium, Aug. 2009.

[19] Enev, Miro, Gupta, Sidhant, Kohno, Tadayoshi, and Patel, Shwetak. Televisions, video privacy, and powerline electromagnetic interference. In *Proc. ACM Conference on Computer and Communications Security* (Oct. 2011).

[20] Eriksson, Joakim, Dunkels, Adam, Finne, Niclas, Österlind, Fredrik, and Voigt, Thiemo. Mspsim–an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session* (2007).

[21] Esmaeilzadeh, Hadi, Blem, Emily, St. Amant, Renee, Sankaralingam, Karthikeyan, and Burger, Doug. Dark silicon and the end of multicore scaling. In *Proc. ACM International Symposium on Computer Architecture (ISCA)* (2011).

[22] Falliere, Nicolas, Murchu, Liam O, and Chien, Eric. W32.Stuxnet dossier. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf, Feb. 2011.

[23] Farag, Sharif, and Srour, Ben. Improving power efficiency for applications. http://blogs.msdn.com/b/b8/archive/2012/02/07/improving-power-efficiency-for-applications.aspx, Feb. 2012.

[24] Faris, Thomas H. *Safe and Sound Software: Creating an Efficient and Effective Quality System for Software Medical Device Organizations*. Asq Press, 2006.

[25] Federal Communications Commission. Code of Federal Regulations, Title 47, Part 15, Sections 101–103, Oct. 2010.

[26] Fredrikson, Matt, Jha, Somesh, Christodorescu, Mihai, Sailer, Reiner, and Yan, Xifeng. Synthesizing near-optimal malware specifications from suspicious behaviors. In *Proc. IEEE Symposium on Security and Privacy* (May 2010).

[27] Freedman, Michael J., Freudenthal, Eric, and Mazières, David. Democratizing content publication with Coral. In *USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)* (Mar. 2004).

[28] Gandolfi, Karine, Mourtel, Christophe, and Olivier, Francis. Electromagnetic analysis: Concrete results. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (May 2001).

[29] Gupta, Sidhant, Reynolds, Matthew S., and Patel, Shwetak N. ElectriSense: Single-point sensing using EMI for electrical event detection and classification in the home. In *Proc. ACM International Conference on Ubiquitous Computing (UbiComp)* (Sept. 2010).

[30] Hall, Mark, Frank, Eibe, Holmes, Geoffrey, Pfahringer, Bernhard, Reutemann, Peter, and Witten, Ian H. The WEKA data mining software: An update. *SIGKDD Explorations 11*, 1 (2009).

[31] Hart, G. W. Nonintrusive appliance load monitoring. *Proceedings of the IEEE 80*, 12 (1992), 1870–1891.

[32] Hart, George W. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine* (June 1989).

[33] Hintz, Andrew. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies Workshop* (Apr. 2002), Roger Dingledine and Paul Syverson, Eds., Springer-Verlag, LNCS 2482.

[34] Hsu, Chih-Wei. Multi-label classification. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#multi_label_classification, Nov. 2011.

[35] Intel Corporation. ATX / ATX12V power supply design guide. http://www.formfactors.org/developer%5Cspecs%5CATX_ATX12V_PS_1_1.pdf, Aug. 2000.

[36] Intel Corporation. ATX specification - version 2.2. http://www.formfactors.org/developer%5Cspecs%5Catx2_2.pdf, 2004.

[37] Internet Archive. Internet archive wayback machine. http://archive.org/web/web.php, Loaded Mar. 2013.

[38] Jana, Suman, and Shmatikov, Vitaly. Abusing file processing in malware detectors for fun and profit. In *Proc. IEEE Symposium on Security and Privacy* (May 2012).

[39] Kasper, Timo, Oswald, David, and Paar, Christof. EM side-channel attacks on commercial contactless smartcards using low-cost equipment. In *Workshop on Information Security Applications*. Aug. 2009.

[40] Khan, Mohammad Maifi Hasan, Le, Hieu K., LeMay, Michael, Moinzadeh, Parya, Wang, Lili, Yang, Yong, Noh, Dong K., Abdelzaher, Tarek, Gunter, Carl A., Han, Jiawei, and Jin, Xin. Diagnostic powertracing for sensor node failure analysis. In *Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (Apr. 2010).

[41] Khronos Group. Opencl — the open standard for parallel programming of heterogeneous systems. http://www.khronos.org/opencl/, Loaded June 2012.

[42] Kim, Hahnsang, Smith, Joshua, and Shin, Kang G. Detecting energy-greedy anomalies and mobile malware variants. In *Proc. International Conference on Mobile Systems, Applications, and Services (MobiSys)* (June 2008).

[43] Kocher, Paul. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology (CRYPTO)* (Aug. 1996).

[44] Kocher, Paul, Jaffe, Joshua, and Jun, Benjamin. Differential power analysis. In *Advances in Cryptology (CRYPTO)* (Aug. 1999).

[45] Koomey, Jonathan G., Berard, Stephen, Sanchez, Marla, and Wong, Henry. Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing 33* (2011), 46–54.

[46] Kramer, Daniel B., Baker, Matthew, Ransford, Benjamin, Molina-Markham, Andres, Stewart, Quinn, Fu, Kevin, and Reynolds, Matthew R. Security and privacy qualities of medical devices: An analysis of FDA postmarket surveillance. *PLoS ONE 7*, 7 (July 2012), e40200.

[47] Krishnan, Srinivas, and Monrose, Fabian. DNS prefetching and its privacy implications: When good things go bad. In *Proc. USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)* (Apr. 2010).

[48] Kuhn, Markus G. Electromagnetic eavesdropping risks of flat-panel displays. In *Workshop on Privacy Enhancing Technologies* (May 2004).

[49] Kuhn, Markus G. Security limits for compromising emanations. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Aug. 2005).

[50] Kuhn, Markus G., and Anderson, Ross J. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding* (Apr. 1998).

[51] LeMay, Michael, and Gunter, Carl A. Cumulative attestation kernels for embedded systems. *IEEE Transactions on Smart Grid 3*, 2 (June 2012).

[52] Liberatore, Marc, and Levine, Brian Neil. Inferring the source of encrypted HTTP connections. In *Proc. ACM Conference on Computer and Communications Security (CCS)* (Oct. 2006).

[53] Liu, Lei, Yan, Guanhua, Zhang, Xinwen, and Chen, Songqing. Virusmeter: Preventing your cellphone from spies. In *Recent Advances in Intrusion Detection (RAID)* (Sept. 2009), vol. 5758 of *Lecture Notes in Computer Science*.

[54] Lu, Liming, Chang, Ee C., and Chan, Mun C. Website fingerprinting and identification using ordered feature sequences. In *Proc. European Conference on Research in Computer Security (ESORICS)* (Berlin, Heidelberg, 2010), Springer-Verlag, pp. 199–214.

[55] McLaughlin, Stephen, and McDaniel, Patrick. SABOT: Specification-based payload generation for programmable logic controllers. In *Proc. ACM Conference on Computer and Communications Security (CCS)* (Oct. 2012).

[56] Microsoft Corporation. Windows. https://www.facebook.com/windows/posts/155741344475532, Jan. 2011.

[57] Microsoft Corporation. Threats at home and work. http://blogs.technet.com/b/mmpc/archive/2013/04/22/threats-at-home-and-work.aspx?Redirected=true, Apr. 2013.

[58] Millikan, Robert Andrews, and Bishop, Edwin Sherwood. *Elements of electricity: A practical discussion of the fundamental laws and phenomena of electricity and their practical applications in the business and industrial world.* American Technical Society, 1917.

[59] Mohan, Sibin, Bak, Stanley, Betti, Emiliano, Yun, Heechul, Sha, Lui, and Caccamo, Marco. S3A: Secure system simplex architecture for enhanced security of cyber-physical systems. *CoRR abs/1202.5722* (Feb. 2012).

[60] Molina-Markham, Andres, Shenoy, Prashant, Fu, Kevin, Cecchet, Emmanuel, and Irwin, David. Private memoirs of a smart meter. In *ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys 2010)* (Nov. 2010).

[61] Murdoch, Steven J. Hot or not: Revealing hidden services by their clock skew. In *Proc. ACM Conference on Computer and Communications Security (CCS)* (Oct. 2006).

[62] P3 International. P3 — Kill A Watt. http://www.p3international.com/products/special/P4400/P4400-CE.html, Loaded Feb. 13, 2012.

[63] Patel, Shwetak N., Robertson, Thomas, Kientz, Julie A., Reynolds, Matthew S., and Abowd, Gregory D. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)* (Sept. 2007), vol. 4717 of *Lecture Notes in Computer Science*.

[64] Popović, Radivoje S. *Hall effect devices*. Taylor & Francis, 2004.

[65] Ransford, Benjamin, Sorber, Jacob, and Fu, Kevin. Mementos: System support for long-running computation on RFID-scale devices. In *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Mar. 2011).

[66] Schweitzer Engineering Laboratories, Inc. *SEL-3354 Embedded Automation Computing Platform: Instruction Manual*, Jan. 2011.

[67] Sherrill, Lynette. Medical device infection data. Personal communication, Jan. 2012.

[68] Song, Dawn Xiaodong, Wagner, David, and Tian, Xuqing. Timing analysis of keystrokes and timing attacks on ssh. In *Proc. USENIX Security Symposium* (Aug. 2001).

[69] Stemen, Pat. Building a power-smart general-purpose windows. http://blogs.msdn.com/b/b8/archive/2011/11/08/building-a-power-smart-general-purpose-windows.aspx, Nov. 2011.

[70] Sun, Qixiang, Simon, Daniel R., Wang, Yi-Min, Russell, Wilf, Padmanabhan, Venkata N., and Qiu, Lili. Statistical identification of encrypted web browsing traffic. In *Proc. IEEE Symposium on Security and Privacy* (May 2002).

[71] Symantec Corporation. Malicious code trends. https://www.symantec.com/threatreport/topic.jsp?id=malicious_code_trends&aid=top_malicious_code_families, Loaded July 2012.

[72] Talbot, David. Computer viruses are "rampant" on medical devices in hospitals. http://www.technologyreview.com/news/429616/computer-viruses-are-rampant-on-medical-devices-in-hospitals/, Oct. 2012.

[73] United States Environmental Protection Agency. ENERGY STAR program requirements for computers. http://www.energystar.gov/ia/partners/prod_development/revisions/downloads/computer/Version5.0_Computer_Spec.pdf, July 2009.

[74] U.S. Department of Homeland Security. ICS-ALERT-12-046-01A—Increasing threat to industrial control systems, Oct. 2012.

[75] U.S. Food and Drug Administration. Cybersecurity for medical devices and hospital networks: FDA safety communication. http://www.fda.gov/Safety/MedWatch/SafetyInformation/SafetyAlertsforHumanMedicalProducts/ucm357090.htm, June 2013.

[76] Valdes, Alfonso, and Cheung, Steven. Communication pattern anomaly detection in process control systems. In *Proc. IEEE Conference on Technologies for Homeland Security* (May 2009).

[77] van Eck, Wim. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers and Security 4* (Dec. 1985), 269–286.

[78] Vuagnoux, Martin, and Pasini, Sylvain. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proc. USENIX Security Symposium* (Aug. 2009).

[79] Walton, Jarred. The AMD trinity review (A10-4600M): A new hope. http://www.anandtech.com/show/5831/amd-trinity-review-a10-4600m-a-new-hope, May 2012.

[80] Wang, Avery. An industrial strength audio search algorithm. In *Proc. International Conference on Music Information Retrieval ISMIR* (2003), vol. 3.

[81] Western Digital Corporation. WD caviar green. http://wdc.com/en/products/products.aspx?id=120, Loaded June 2012.

[82] White, A.M., Matthews, A.R., Snow, K.Z., and Monrose, F. Phonotactic reconstruction of encrypted VoIP conversations: Hookt on Fon-iks. In *Proc. IEEE Symposium on Security and Privacy* (May 2011).

[83] Wright, Charles V., Ballard, Lucas, Monrose, Fabian, and Masson, Gerald M. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *Proc. USENIX Security Symposium* (2007).

[84] Yee, Bennet, Sehr, David, Dardyk, Gregory, Chen, J. Bradley, Muth, Robert, Ormandy, Tavis, Okasaka, Shiki, Narula, Neha, and Fullagar, Nicholas. Native client: A sandbox for portable, untrusted x86 native code. In *IEEE Symposium on Security and Privacy* (2009), pp. 79–93.