Current Events: Identifying Webpages by Tapping the Electrical Outlet

Shane S. Clark[1], Benjamin Ransford[1], Jacob Sorber[2], Wenyuan Xu[3], Erik Learned-Miller[1], and Kevin Fu[1]

[1]University of Massachusetts Amherst
[2]Dartmouth College
[3]University of South Carolina

Abstract

Computers plugged into power outlets leak information by drawing variable amounts of power when performing different tasks. This work examines the extent to which this side channel leaks private information about web browsing. Using direct measurements of the AC power consumption with an instrumented outlet, we construct a classifier that correctly identifies unlabeled webpage-activity traces from a set of 50 candidates with 87% precision, 74% recall, and 99% accuracy. The classifier rejects samples of 441 pages outside the corpus with a false positive rate of less than 2%. It is also robust to a number of variations in webpage loading conditions, including encryption. Characterizing the AC power side channel may help lead to practical countermeasures that protect user privacy from untrusted power infrastructure.

1 Introduction

Web browsers increasingly take advantage of hardware acceleration to implement rich user interfaces. For example, the HTML5 standard defines a generic `canvas` element to support flexible graphics [40], so browsers provide GPU-accelerated support for canvas operations [1]. Other examples include offline storage, streaming video, client-side scripting, and native-code execution [43]. These mechanisms allow browsers to use more of the available hardware to provide rich user experiences, stressing components in potentially new ways.

In parallel with the trend toward hardware acceleration of web content, the dynamic range of components' power consumption is increasing. Modern chips often draw more power than their predecessors when active, but thanks to ongoing energy-efficiency initiatives such as ENERGY STAR [37], they support aggressive power management and spend an increasing fraction of their time in low-power idle states.

Richer interfaces, greater hardware use and greater dynamic range mean that a computer's power consumption while loading webpages fluctuates more than it may have in the past. A natural question is whether *different* webpages induce *identifiable* patterns of power consumption via their different use of computing resources. It may seem intuitive that a webpage simultaneously playing 100 videos probably requires more power than a 100-byte plain text webpage. In practice, several challenges complicate the task of identifying webpages:

- Competing websites often imitate one another, resulting in similar resource-use patterns.
- Most websites optimize for load time, so many popular webpages load in approximately the same amount of time.
- A complex software stack sits between a webpage and the power supply, introducing layers of indirection that may buffer or hide fine-grained events.
- Many websites change frequently, often presenting different content (e.g., ads) to different users.

This paper experimentally demonstrates that, despite the above challenges, different webpages *do* induce different power-consumption patterns that can be used to identify them. Recent work on AC power analysis shows that a classifier trained on 15-minute HDTV video segments can identify unlabeled 15-minute clips with over 96% accuracy [12]. Our work advances the state of the art by using a different measurement and classification strategy to classify 15-*second* traces of a personal computer's power consumption, identifying webpages instead of video clips. It conceptually bridges the gap between previous work on circuit-level direct-current (DC) power analysis [23] and household activity recognition via single-point alternating current (AC) power measurement [16, 34, 15].

This paper's primary contribution is a collection of empirical results showing that potentially sensitive information leaks via an AC power side channel. To our knowledge, this work is the first to use AC power analysis to identify web traffic. Using a covertly modified electrical outlet to record power consumption, we trained and tested a classifier with over 50 hours of traces representing more than $8,500$ webpage loads from a set of 50 websites representing over 30% of global page views. Given an unlabeled trace of a webpage in the training set, the classifier identifies the correct match with 87% precision (resistance to false positives) and 74% 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%. Additionally, the classifier is robust against a variety of potentially confounding influences, including the use of an encrypting VPN. We also suggest and evaluate several countermeasures to mitigate the threat of information disclosure.

Threat model. This paper focuses on the privacy implications of an AC power side channel, and does not consider an attacker who manipulates the power line to intentionally induce patterns. This work considers an attacker who has physical access to a power outlet the victim might use, but not necessarily to the victim's network. Such a power outlet might be in a coffee shop, airport, commercial office, hotel-room wall, or home. With a time window of only a few seconds, an attacker can unscrew the outlet's faceplate, tap two wires, and replace the faceplate. Alternatively, an attacker might slip a lookalike faceplate over the target outlet, akin to ATM skimming attacks described elsewhere [32]. Or an attacker posing as a friendly traveler might offer an outlet on a power strip to the victim. A final step is to connect a logging back end to the instrumented outlet. The experiments in this paper use a 250 kHz sample rate, which with 16-bit samples generates 4 Mbps, within reason for a modern cellular network or flash-memory data logger.

2 Information Leakage via Variable AC Power Consumption

Computer components require DC power, but most power systems worldwide (including in the U.S.) deliver AC power to customers. To convert AC power to DC power, a computer relies on a switched-mode power supply (SMPS). An SMPS uses a rectifier in concert with a switching voltage regulator and filtering capacitors to produce a steady DC voltage output. The power-conversion components are typically inside a fan-cooled box in a desktop chassis, a "brick" on a laptop's power cord, or a "wall wart" at the end of a mobile-phone charging cable.

Loads and power. An SMPS is considered a *nonlinear load* because its capacitive and inductive components distort the shape of the AC source's sinusoidal current waveform, drawing the current and voltage waveforms out of phase and returning power to the source. The total power passing through the SMPS in both directions is called the *apparent power*, P_{apparent}, and is measured in volt-amperes (VA). The power used to perform work in the load is known as *real power*, P_{real}, and is measured in watts (W). The *power factor* is the ratio of real power to apparent power, $PF = P_{\text{real}}/P_{\text{apparent}}$, and measures how efficiently a device is consuming power. Resistive *linear* loads, like an incandescent light bulb, have power factors very close to 1.0, but nonlinear loads like SMPSes, which store and return some of the apparent power as *reactive power*, have lower power factors and distort the source waveform more dramatically, resulting in information leakage.

2.1 Causes of Information Leakage

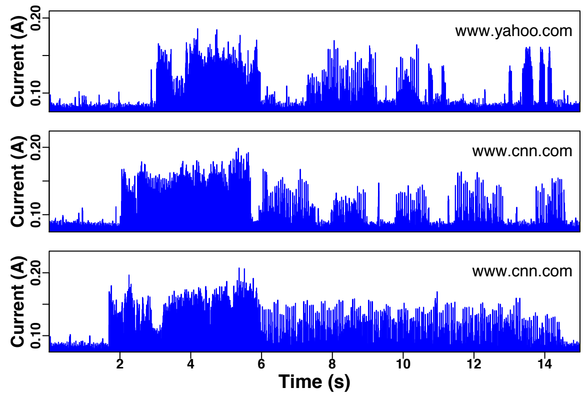
An SMPS leaks information in at least three ways.

First and foremost, its AC power consumption is necessarily correlated with the sum of the components' DC power consumption. If any particular component conveys information via its power consumption, then that information may appear in an AC power trace. This intuition is the basis for our analysis. Our approach, sampling the current on a physical wire in the AC power line, is the most direct way to look for information in AC power consumption patterns.

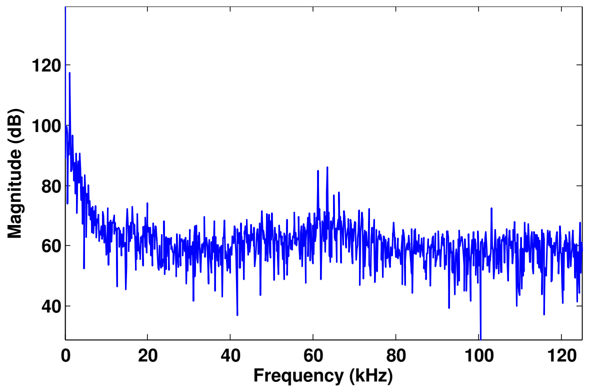
Second, as mentioned above, an SMPS leaks information via reactive power. When the computer executes tasks, the DC load varies, so the AC load varies correspondingly. As mentioned above, the load distorts the AC voltage and current waveforms out of phase, generating reactive power. The key observation is that the amount of reactive power varies along with the load.

Third, like any electrical device that switches on and off, an SMPS emits electromagnetic interference (EMI) that other devices can detect. We refer the reader to Kuhn for in-depth exploration of EMI [26]. To meet emissions standards (e.g., FCC Title 47 CFR Part 15 in the U.S. [13]), SMPSes contain inductors that filter out noise at frequencies above the voltage regulator's switching frequency (which varies from unit to unit). The remainder of this paper demonstrates that this EMI filtering—presumably designed without information leakage in mind—does not prevent sensitive information from appearing on the AC power line.

This paper extends the recent results of Enev et al., who studied SMPS EMI in isolation [12], by including information from all three of these leakage sources, allowing classification with shorter inputs (seconds, versus minutes).



(a) Time-domain plots of AC current consumption as a 2008 MacBook loads two different pages. Despite the lack of obviously characteristic information in the time domain, the classifier correctly identifies all of the above traces.



(b) Frequency-domain plot of AC voltage as a 2008 MacBook loads the front page of `cnn.com`.

Figure 1: Time- and frequency-domain plots of several power traces in our classifier's training set.

2.2 Analysis in the Frequency Domain

Our traces are sequences of voltage measurements taken at a fixed sample rate (250 kHz). Plotting a trace in the *time domain* (Figure 1a) reveals a pattern of peaks that intensify during periods of activity. Time-domain analysis enables a classifier to reason about history and duration, and is useful for analyzing network traffic (e.g., encrypted VoIP streams [41]) and other well-formed inputs. Time-domain plots are also easy to inspect visually in simple cases. However, to prepare traces for classification, we use a Fourier transform (Figure 1b) to represent the entire trace in the *frequency domain* in terms of the component frequencies that sum to the trace waveform, then analyze the entire trace at once. This approach offers three important advantages over time-domain analysis: (1) We can meaningfully compare traces of different lengths; (2) we can remove periodic signals such as the main AC frequency (60 Hz in the U.S.) or predictable noise; and (3) the additive nature of the SMPS's power consumption preserves frequency information from individual components' power consumption.

3 Related Work

AC power event recognition. This work focuses on classifying short-lived run-time events 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 [16, 17]. 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 [34, 15]. By 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.

Our classifier's inputs are traces of computer power consumption, in contrast 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 and roughly 11% accuracy with the other two [12]. To increase accuracy, they used an isolating transformer and a high-pass filter to remove noisy low-frequency signals and measured EMI at a sample rate of 500 kHz, and also classified signals (video segments) more than 15 minutes long. In comparison, we focus on classifying signals containing shorter periods of activity; our longest webpage access lasted less than five seconds. Our sensing apparatus is somewhat simpler, relying instead on a single sense resistor and no hardware filtering.

This work extends a recent exploration of AC power analysis that classified among 8 webpages with a time-domain classifier that was too computationally intensive to operate in real-time [8]. Our work tests a total of 491 popular webpages in the frequency domain, resulting in better classification performance, and our supervised learning algorithm permits classification among 50 candidates in nearly real-time (over $3,400$ trace classifications per second).

Network traffic analysis. 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 [18, 28, 30, 36]. Our classifier uses AC power traces as input rather than network traces, under the assumption that we cannot automatically determine packet characteristics that would be useful for a network-trace classifier.

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 [41]. Their classifier considered VoIP packet lengths in the time domain and combined several probability models using Bayesian inference. Perhaps most surprisingly, the classifier of White et al. works even on encrypted traffic; our classifier shares this property (§5.3).

We borrow a countermeasure from past work on keystroke timing attacks. Song et al. observed that the SSH protocol (in 2001) preserved inter-keystroke timing information that, when provided to a trained model, reduced the search space for brute-force password cracking by a factor of 50 [35]. They suggested the countermeasure of sending packets at a constant rate regardless of whether there is data to send; we adapt this strategy to power consumption to confound a frequency-domain classifier like ours.

Parasitic modulation. 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 [38]. Kuhn further analyzed leakage from CRT and LCD monitors based on parasitic modulation [25, 26, 27]. 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 [39]. 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 [9].

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 [22, 23]. 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. Combining wireless and power analysis, Kasper et al. implemented an attack against contactless smartcards based on changes in the bandwidth of the carrier signal supplied by a reader in response to power consumed by a smartcard [20].

4 Supervised Learning with Support Vector Machines

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 make the reasonable assumption 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.

We use a *supervised* learning mechanism because we know the correct label for each sample during training. Specifically, we train *support vector machines* (SVMs) using the open-source library libsvm [10]. Given a training set of labeled inputs, an SVM creates a *hyperplane* in n dimensions (where n is the number of features) that maximally separates positive examples from negative examples in the n-dimensional feature space.

4.1 Training Phase

Feature selection. Classification involves choosing *features*—salient properties that may differ from sample to sample—on which to train a classifier. A straightforward classifier could simply consider the *length* feature of a sample in the time domain, defined as the length of time for which power consumption is high between two idle periods. 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. Mean, minimum, and maximum power are equally unsuitable choices for similar reasons. A more robust approach is to classify traces based on features from the frequency domain, as explained in Section 2.2.

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. (Section 5.3 presents the results of varying the sample rate.)

Training binary classifiers. Suppose we wish to train our classifier to recognize the home page of `example.org`. An SVM is a *binary* classifier, so it needs labeled positive (*this is `example.org`*) and negative (*this is not `example.org`*) samples. Some or all of the positive samples look similar in the frequency domain, and therefore naturally form a cluster in the feature hyperspace. The negative samples ideally do not cluster with the positive samples, in which case the SVM learns to separate them. We train one SVM for each label—i.e., for each of the 50 webpages in our data set.

4.2 Classification Phase

After training all 50 SVMs, we ask 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 falls on the positive or negative side of its hyperplane, thereby generating a yes or no answer as to whether the test sample was an instance of the webpage it was trained to recognize. We do not implement a multi-class labeling solution in which all 50 SVMs collectively generate a single output, but there are a variety of well-studied techniques for this purpose and libsvm implements several of them [19].

There are three notable details of the training process. First, we use input scaling to prevent features with larger numeric ranges from dominating those with smaller ranges. Second, we use 10-fold cross-validation, which repeatedly splits the training set into training and testing subsets. By repeatedly training and testing on subsets of the training data and modifying the model based on feedback, cross-validation ensures that large numbers of similar examples do not cause overfitting. Finally, we use a radial basis function (RBF) kernel, as recommended by the authors of libsvm [10], to construct a statistically consistent classifier.

5 Evaluation

This section summarizes our evaluation methods and experimental results over a wide range of conditions. We aim to answer the following questions:

- How effectively can the SVM classifier differentiate webpages from one another?
- What sampling rates result in good classification?
- How does the classifier's performance change as the size or diversity of the training set changes?
- How robust is the classifier in the presence of network traffic anonymizers, content distribution services, encryption, and caching, as well as changes in operating system, hardware, or network interface?
- How does the classifier's performance change when the test traces include background activities?
- How well does the classifier correctly exclude samples of pages outside the corpus?

We find that our classifier can differentiate webpages with high precision and recall rates (averaging 87% and 74%, respectively) and that it is robust against many of the variations we tested, including the use of a VPN, but not including changes of machine or operating system. Where our classifier performs poorly, we find in most cases that increasing the training set's size and diversity improves its performance along all metrics. The total number of power traces we tested across all experiments was $8,585$, chronicling over 50 hours (160 GB) of 250 kHz trace recordings.

5.1 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 (by audience size) [7], discarding duplicates and adult websites. By Alexa's estimates, these top 48 websites represent over 30% of global page views (Figure 2). 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 the first author's department's home page, a < 1 ms round trip from our measurement point, bringing the number of webpages in our training set to 50.

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 50 websites. To record how each webpage might appear over the course of a typical day, we used a custom Chrome extension (see §5.2) to collect at

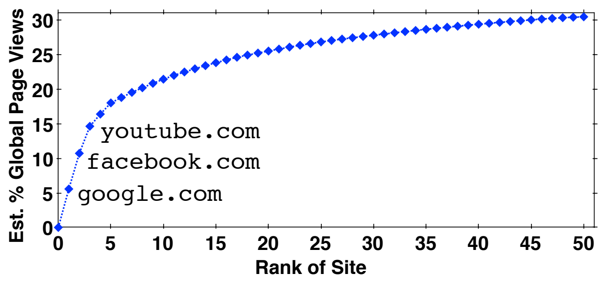


Figure 2: The top 50 websites according to Alexa in terms of percentage of global estimated page views.

least 90 consecutive traces of each. 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 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.

Evaluation metrics. Because our classifier uses standard machine-learning techniques, we use standard metrics from machine learning to evaluate its performance. 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—$(tp+tn)/(tp+tn+fp+fn)$, the fraction of all examples that are correctly labeled—is easy to game. In most of our experiments, the number of negative examples is roughly 49 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, however, there is a tradeoff between precision and recall. While it is trivial to achieve either 100% precision or 100% recall by rejecting or accepting all examples respectively, a classifier can maximize both only by reaching 100% accuracy.

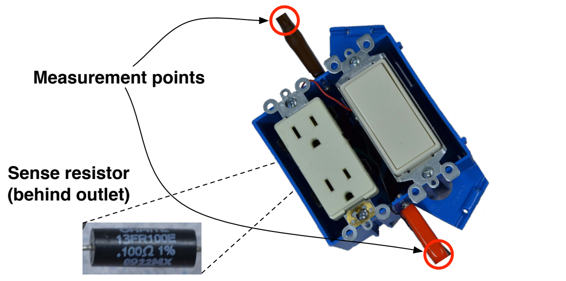


Figure 3: We captured AC voltage traces by instrumenting a standard North American NEMA 5 AC power outlet. We placed a 0.1Ω, 10 mm sense resistor (enlarged on left) in series with one terminal of the outlet, plugged a computer into the outlet, and sampled the resistor's voltage at 250 kHz with a data acquisition unit.

5.2 Experimental Setup

This section describes the experimental setup we used to capture AC power traces.[1]

Safety note: This paper is not a complete manual for electrical safety. Measuring "hot" terminals is potentially fatally dangerous and should be conducted only under qualified supervision. Do not try this at home.

Using an instrumented outlet (described next), we measured the power consumption of a 13-inch Apple MacBook, circa 2008, with a dual-core Intel Core 2 Duo processor, 4 GB of RAM, Intel GMA X3100 GPU, 80 GB Corsair SATA II MLC solid-state drive, and Mac OS 10.6.8, from which we removed the battery. While the majority of our experiments are based on power traces gathered from the MacBook, we also explored differences among hardware and software configurations. To this end, we gathered traces using a Dell Vostro desktop PC with a quad-core Intel Core i5 processor, 4 GB of RAM, AMD Radeon 6450 GPU, and 250 GB Seagate 7200 RPM magnetic hard disk. We tested the desktop PC under Windows 7 and Ubuntu 10.04. 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.

[1]We use the terms *power trace*, *voltage trace*, and *current trace* interchangeably. What we actually record is a *voltage trace* that maps trivially to current ($I_{sense} = V_{sense}/R_{sense}$, with R_{sense} constant) and therefore power ($P = I_{sense}V_{RMS}$, with V_{RMS} a constant 120 volts (RMS) in North America).

To record each workload's power signature, we monitored electrical current between the power supply and an instrumented outlet. A modern AC outlet has three terminals: *hot*, *neutral*, and *ground*. To measure a power supply's instantaneous current on the hot–neutral circuit, we placed a 0.1 Ω *sense resistor* (part #13FR100E-ND), about 10 mm from terminal to terminal, in series with the hot terminal of a standard outlet. 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. Figure 3 depicts our instrumented 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. We chose to sample at 250 kHz because it was the maximum rate supported by the DAQ and we had no prior knowledge of which frequency ranges would contain identifying information.

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 [5] to purge the browser environment of confounding data. To compare webpage identifiability across browsers, we also used the iMacros extension for Firefox [4] 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.

5.3 Classification Results

Page differentiation. Our SVM classifier effectively differentiates among the 50 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 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.

After gathering ∼90 traces for each of the 50 webpages on the MacBook, we used the experimental protocol described in Section 4.2 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 4,638 traces. We used roughly half of these traces for training and the remaining traces for testing. Figure 4 plots precision and recall for each of the 50 webpage labels. With ∼45 training examples per label, the SVM classifier achieves an average 87% precision and 74% recall over all webpages.

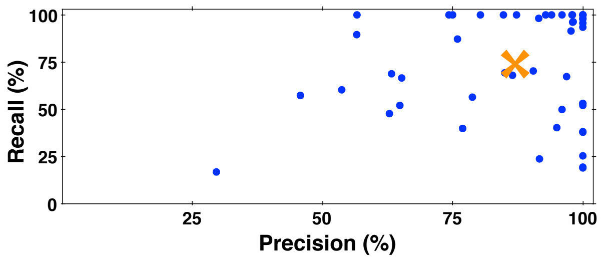


Figure 4: Precision and recall for the 50 webpage labels. The arithmetic mean over all webpages is plotted as a large X. The SVM classifier achieves 100% precision and recall for 5 of the 50 webpages. The outlying webpage label with the lowest precision and recall is `godaddy.com`, a pathology we believe is due to its heavy use of dynamic content and large download size.

The classifier's performance varied among the tested webpages. The classifier maintained 100% precision and recall for 5 of the 50 webpages, and at least 90% precision and recall for 18 of the 50 webpages. In contrast, the classifier's precision and recall for the worst page, `godaddy.com`, were only 30% and 17% respectively. A plausible explanation for the classifier's poor performance on `godaddy.com` is that it loads a wide variety of ads and dynamic content, resulting in different resource-use patterns each time it loads. Encouragingly, experimental results suggest that increasing the size of the training set would narrow the performance gap by improving classification performance for the most difficult-to-classify webpages.

Sampling rate differentiation. Decreasing the sampling rate at which an instrumented outlet records AC voltage—to enable trace exfiltration via a low-throughput mobile phone, for example—decreases the SVM classifier's performance along our metrics, but not as much as one might expect. To understand how robust the SVM 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.

Table 1 compares the results with the original sampling rate against results with simulated lower sampling rates. A reduction to 100 kHz yields nearly identical re-

Sampling Rate	Precision	Recall
2 kHz	58.29%	50.88%
12 kHz	78.72%	65.77%
100 kHz	86.78%	73.70%
250 kHz	86.75%	74.00%

Table 1: The SVM classifier's performance degrades as we decrease the sampling rate in simulation, but precision and recall remain over 50% even with a hundredfold reduction in the original 250 kHz sampling rate, suggesting that simple tracing hardware and low-bandwidth exfiltration (via, e.g., mobile phone) are feasible.

sults, but the SVM classifier's performance degrades as the simulated sampling rate decreases further. Reducing the sampling rate by a factor of more than 20 (from 250 kHz to 12 kHz) incurs only a 10% reduction in average precision and recall. These results suggest that the low-pass filters in commercial SMPSes are not sufficient to prevent information leakage, but that more aggressive filtering could *reduce* information leakage. Section 6 discusses a countermeasure based on this observation.

Training set size. Increasing the size of the training set from ∼45 to ∼250 and then to ∼450 improved the classifier's performance on the most difficult-to-classify webpage, `godaddy.com`. With only ∼45 training samples, the SVM's precision and recall for `godaddy.com` were 30% and 17% respectively. The additional training examples dramatically improved both, raising the precision to 75% and the recall to 59%. We also evaluated an even larger training set, with 477 samples, which improved the precision and recall to 80% and 100% respectively. This result suggests that one can improve the SVM classifier's performance by increasing the size of the training set.

Diverse browsing conditions. We varied the conditions under which our browsers 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 [14].

Our training and testing setup was as follows. We chose a single webpage from our set of 50, then gathered the following sets of traces on our MacBook:

- **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. For example, to study 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 used for the webpage-differentiation experiment. In contrast to training only on the *Base* set, this training approach avoids overfitting the SVM to that set. To illustrate the effects of overfitting, we trained the classifier using only the *Base* set for `google.com` and then tested using the *VPN* set for `google.com`. Because of overfitting, the classifier rejected all samples from the *VPN* set.

After training the classifier, we instructed it to classify the traces in the test set. We repeated this process for three webpages: one simple page (`google.com`) for which the classifier achieved 100% precision and recall in the page-differentiation experiment; one complex page (`cnn.com`) with poor precision (57%); and one complex page (`cnet.com`) with poor recall (25.5%). Table 2 shows the results, which we summarize below.

For `cnn.com`, this training approach achieved much better precision on each of the test sets than it did in the page-differentiation experiment. In the case of `cnet.com`, the additional training diversity radically improved recall, from 25.5% to 89.6%. 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. 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*

Condition	Precision	Recall
`google.com`		
Time	100.0%	100.0%
WiFi	100.0%	100.0%
VPN	100.0%	100.0%
Cache	100.0%	100.0%
CDN	100.0%	10.0%
`cnn.com`		
Time	90.3%	100.0%
WiFi	90.0%	99.4%
VPN	90.3%	99.4%
Cache	90.3%	99.4%
CDN	88.9%	85.6%
`cnet.com`		
Time	100.0%	89.6%
WiFi	100.0%	89.6%
VPN	100.0%	89.6%
Cache	100.0%	89.6%
CDN	100.0%	89.6%

Webpage	Precision	Recall
`google.com`	100.0%	100.0%
`cnn.com`	56.6%	100.0%
`cnet.com`	100.0%	25.6%

Table 2: Classification performance under diverse browsing conditions (top table): Training the classifier under different conditions increased the diversity of traces it could correctly label. The bottom table lists baseline precision and recall from the page-differentiation experiment.

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.

Operating system diversity. Training on traces from a single operating system appears to overfit the classifier to that operating system. However, training on traces from two operating systems allows the classifier to correctly identify both kinds. 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. All of these differences may play leading 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.

Machine diversity. 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 10.6.8) and tested on the desktop PC (running Windows 7), then switched the roles and trained and tested again. For both `google.com` and `cnn.com`, the SVM failed to correctly label any traces from one machine when trained only on examples from the other.

Training on examples from *both* machines allowed the SVM to classify traces from both machines accurately: 100% precision and 100% recall for `google.com`, and 67.3% precision and 75.5% recall for `cnn.com`. This result suggests that, as in the operating system experiment, the problem may lie in the lack of training diversity. In the future, we intend to test this hypothesis by training an SVM on a wide range of machines and then testing traces from machines that are not represented in the training set. It may be possible to parameterize differences among hardware configurations and automatically adjust traces to improve classification performance.

Background webpages. Noting the tendency of users to browse multiple websites at the same time, we chose two of the 50 webpages in our training set—`google.com` and `cnn.com`—and loaded them with other webpages already loaded in other tabs, to measure the classifier's sensitivity to background pages. The Chrome extension initiated the target page load with the background page already open and fully loaded. While concurrent page loads would have created more background power consumption, we consider it unlikely for most users to browse in this fashion. For each of the two webpages, we gathered a set of traces with each of the two webpages in the background, creating four combinations. For example, we separately tested `google.com` with `google.com` in the background and with `cnn.com` in the background. For each of the four combinations, we trained an SVM on the other three.

For three of the four combinations, the classifier maintained $> 70\%$ recall and 65% precision. The notable exception was the set of traces in which `cnn.com` loaded with `google.com` in another tab. Under this condition, the precision and recall were each 27%. This result is surprising because `cnn.com`'s front page is more complicated, both visually and in terms of power consumption patterns, than `google.com`'s. Once again, the performance would likely improve with a more diverse training set. For these experiments, we used only one set of traces with background pages for training in each case.

Exclusion of unknown pages. A possible explanation for the classifier's ability to discriminate between positive and negative examples is that each SVM clustered negative, rather than positive, examples in its feature space. With training and testing sets that resembled each other, a classifier could perform equally well 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. A high false positive rate would suggest that the SVMs clustered only the negative examples, erroneously learning to positively label samples outside the cluster of negative examples it had learned.

We gathered one trace from each of 441 webpages randomly selected from a list of 1 million popular pages published by Alexa [6], making sure to remove pages that were already in the training set. We then tested all 441 pages against all 50 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.

6 Countermeasures

This section sketches several countermeasures to mitigate the threats described in Section 1. These countermeasures can be broadly classified as: (1) *software* countermeasures that attempt to obscure, obfuscate, or disguise computational workloads in the AC signal to remove the side channel; and (2) *hardware* countermeasures that involve deploying additional electronic components to remove workload artifacts. We do not consider a Sensitive Compartmented Information Facility (SCIF) or other military-grade countermeasures because we believe such approaches are not feasible for civilians.

Hardware and software countermeasures both present inherent tradeoffs. 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.

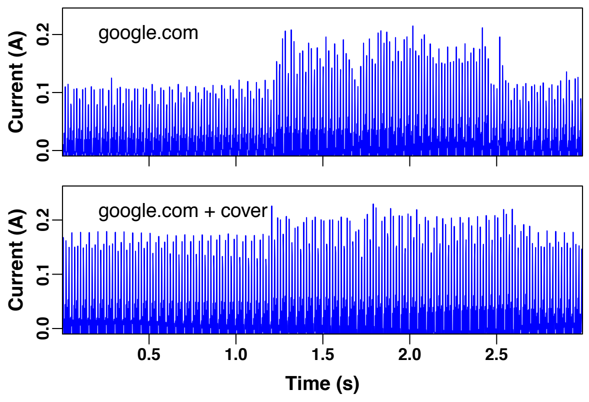


Figure 5: Time-domain traces of `google.com` without (top) and with cover traffic. We plot the absolute value of the current for ease of inspection.

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*. Figure 5 illustrates that cover activity partially conceals a sensitive task in the time domain. The cover activity appears to spread out and distort the shape of the page load activity, although the original signal is still visible. We tested cover activity by loading one processor core completely and recording samples of `google.com`. The classifier misclassified all 41 of the obfuscated samples.

Unless it saturates the available resources by maximizing system-wide power consumption, a cover task that merely adds a constant amount of power consumption might be easy to filter or subtract from the AC signal. The ElectriSense system offers some evidence that this subtraction is feasible. 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 [15]. 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.

Software countermeasure: Delays and throttling. We adopt a defensive idea from Song et al. [35]: 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 5.3) 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.

Software countermeasure: Cloud proxies. Services such as OnLive or Amazon's Silk offload computation (video-game and webpage rendering, respectively) to cloud-based proxies to increase throughput and decrease latency [3, 2]. The use of a similar cloud-based proxy could reduce a client's information leakage by reducing computational work and making webpage loads more closely resemble one another. Such a cloud proxy may also present a privacy risk, but privacy-conscious users may be equipped to evaluate the tradeoff.

Hardware countermeasure: Batteries. To sidestep the AC power side channel, a user could avoid using an AC power connection at all while performing sensitive tasks, instead favoring battery power.

To test under what conditions a laptop with a battery leaked information via AC power, we captured traces from our MacBook when its battery (an OEM MacBook battery with one year of use) was charging and when it was fully charged. When the battery was full but the AC adapter remained plugged in, the laptop's power consumption via the SMPS was indistinguishable from its power consumption with the battery removed (our usual testing condition for the MacBook). Laptops generally prefer to use AC power whenever possible to extend the life of the battery.

To determine whether charging during trace recording would confound the classifier, we repeatedly collected traces of the MacBook's AC connection while charging the battery from empty to full. The idle power consumption with a charging battery was visually indistinguishable from the idle power consumption without a battery, except for an increase of the noise floor. Although we did not test the classifier on the traces we collected while charging the battery, we believe that battery charging is unlikely to affect classification accuracy more than other low-power background activities to which the classifier was robust in our experiments (§5.3).

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 (§2.1). Our experiments reveal that, for the SMPSes we tested, the frequencies useful for classification are below the internal filter's cutoff. An additional low-pass filter (a larger inductor that resists lower-frequency changes in current) could remove information below the existing threshold, but larger inductance to filter out web activity would likely increase the cost and physical size of an SMPS.

7 Discussion

Automatic event separation. Our evaluation used timing information from a Chrome extension to divide long traces into individual page load samples, but automatically identifying regions of interest is straightforward. Based on our measurements of roughly 5 seconds as the maximum load time for webpages in our traces, and our observation that the beginning of webpage activity corresponded to a dramatic spike in power consumption, we implemented a simple algorithm to automatically find regions of interest in the raw AC voltage traces. The algorithm performs the following steps: (1) Take the absolute value of all samples; (2) remove all samples below a tunable threshold (e.g., $9\times$ the mean sample value); (3) downsample the data with averaging to remove transient spikes in current consumption; and (4) consider the first sequence of 10 consecutive non-zero samples to be 1 second from the activity start, extracting a 6-second subsequence of the original time series. We processed 25 traces chosen at random from our training set and manually verified that the algorithm correctly identified a webpage load in each case. This process still requires domain knowledge about how long events last and by how much they may be separated; we did not attempt to develop a generalized event recognizer.

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.

7.1 AC Traces and Power Budgets

To determine which system components have the most impact on power consumption, we used a P3 Kill A Watt power monitor [33] to measure our MacBook's power consumption while stressing different components. Table 3 summarizes the results.

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: MacBook (2008 model) power consumption under various types of load. Numbers beginning with + are relative to the baseline of 8 W.

The MacBook's CPU and GPU dominate power consumption under load. The network interfaces draw comparatively little power. Even when saturated, the WiFi interface consumes only half the power of the SSD and does not approach the CPU or GPU's power consumption. The low power consumption of the network interfaces suggests that AC power analysis reveals system information not discernible via network analysis.

Applicability of network-trace classifiers. A related question is whether one can infer detailed information about network activity by inspecting power traces. If so, it could be feasible to apply existing classifiers such as those that use packet timings [35] or lengths [41, 42] to classify webpages according to their network traffic.

An experiment suggests that an AC power trace does 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 6 shows an example from our tests. The computer consumes power performing other tasks before the network interface actually begins to send and receive packets.

We gathered *timeline* traces using Chrome's developer tools and analyzed them to better understand which aspects of web browsing induce high loads on the computer. The timeline tool logs a sequence of events describing how the browser sends, receives, and processes information. We profiled 10 of the 50 webpages tested in this work and found that so-called *Paint*—screen drawing—events dominated the timelines for all

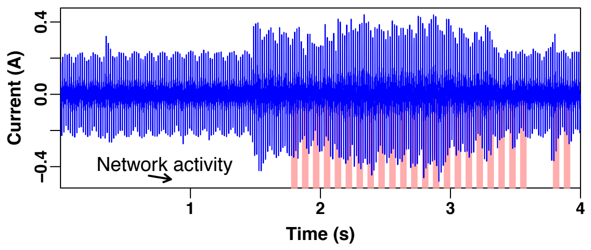


Figure 6: A plot of current consumption (dark) overlaid with network activity (light bars along the bottom). The network activity is correlated with high current consumption, but is not the only cause. Spikes before and after network activity show that local computation dominates the consumption.

of them. The number of Paint events varied among the webpages we profiled, but was always among the most time-consuming event types. We suspect that much of the identifiable information in our traces comes from the number and length of CPU- and GPU-intensive screen-drawing events, which Table 3 supports.

7.2 Future Work and Open Problems

More sophisticated classifiers could yield even better precision and recall than our stable of SVMs. We believe our experimental setup and designs will enable the research community to more rapidly investigate more generalized questions about information leakage via AC power. The following paragraphs outline a variety of open research questions and our plans for extending this work.

Training with more diversity. A weakness of our classifier is that it fares poorly on test samples from a machine when it has been trained only on traces from a single different machine (§5.3). It similarly overfits to a single operating system and fares poorly when tested against traces from another. Both of these problems appear to stem from inadequate diversity in the training set.

In addition to making our hardware design and software publicly available, we intend to build a library of additional traces from a diverse set of machines. Varying machine type, operating system, and network location are likely starting points. We expect this variation will result in gains similar to those we found in our experiments.

Indirect measurements. To date, we have focused on measuring current draw directly, in order to capture as much leaked information in our traces as possible. Alternative measurement techniques—using a Hall effect sensor (described earlier in §7) to measure the current through a wire or measuring radiated electromagnetic emanations (as described by Kuhn [26])—may be less intrusive than directly measuring current. However, while these indirect measurements should provide usable features for our classifier, we expect both information content and classification performance to be lower because of measurement noise.

Another possibility is indirect measurement similar to that of Enev et al. [12]: 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.

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.

Another possible improvement to our frequency-domain analysis is to use a windowed Fourier transform—i.e., transform segments of the time-domain signal—instead of transforming the entire trace. In contrast to our whole-trace approach that dilutes the strength of short-lived high-frequency information, windowing would strengthen higher frequencies and attenuate lower frequencies.

An extension that takes advantage of SMPS load characteristics would be to simultaneously sample both voltage and current. As Section 2 discusses, SMPSes are nonlinear loads that 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 [24, 11] as a user types, the activity creates distinct spikes of CPU activity after each keystroke. Figure 7 illustrates that the spikes are distinct in the time domain. Augmenting the classifier with a URL length estimate could eliminate obviously bad matches.

Detecting other activities. As we have emphasized in this work, computers are complex, general-purpose devices. The space of possible actions that a user might take is vast. While we have focused on web activity in order to address a well-defined, tractable problem, future work could address a broad range of other activities. As Figure 7 demonstrates, detailed information may be available from many sources. Tasks as simple as counting keystrokes, regardless of whether different keys can be recognized, may reveal sensitive information. Song et al. have demonstrated that counting keystrokes can reduce the search space for brute-force password cracking by a factor of 50 [35].

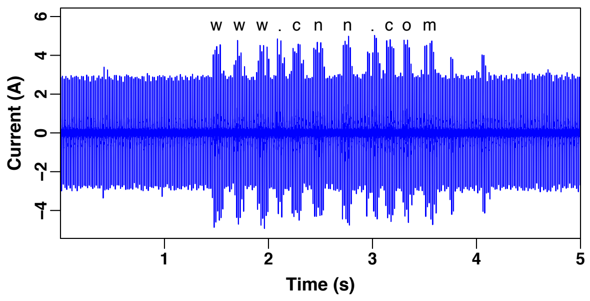


Figure 7: Text boxes that trigger CPU activity with each key press yield extra information on the AC power channel. In this trace from Chrome on our MacBook, it is visually evident how many keys we pressed. Even a rough estimate could be used to inform a webpage classifier.

Beneficial uses of power monitoring. This paper has focused primarily on privacy threats from a power line-monitoring attacker. However, interpreting patterns in power traces can potentially be used for good, to improve system security and performance. Abnormal changes in DC power consumption have already been used to detect some instances of malware in mobile phones [21, 29]; we envision a similarly designed mechanism for general-purpose computers plugged into AC power. Detecting misbehaving software externally has many advantages over more traditional methods; however, desktops, laptops, and servers exhibit a higher degree of software diversity than mobile phones, and building a reliable profile of "normal" computer behavior poses a significant challenge.

Looking beyond computers, increasingly fine-grained power metering of entire buildings is emerging as a critical element of many "smart grid" designs. Inferring behavior patterns from electric power use in homes and office buildings has been proposed as a way to improve algorithms that tailor power generation and distribution to changes in consumer demand, and privacy researchers have already expressed concern about potential privacy violations [31]. In the future, we plan to extend both the classifiers and countermeasures described in this paper to ease the tension between privacy and system utility, allowing companies to optimize their systems with information that homeowners selectively provide.

8 Conclusions

This work shows that a *general-purpose* classifier can identify webpage activity patterns in traces of a computer's AC power consumption. With a data set of over 8,500 power traces of 50 popular webpages, a trained classifier correctly labeled unseen traces with 87% precision and 74% recall. These metrics are robust against several variations including the use of an encrypting VPN. To our knowledge, this is the first paper 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 software techniques that mitigate the risks of using untrusted power infrastructure without foiling advanced power management.

A AC Power as Covert Channel

The classifier's success rate suggests that the AC power channel represents workloads *consistently*. A natural question to ask is whether it is possible to generate *intentional* power signatures by varying the workload in a desired way. We wrote a simple script to amplitude-modulate the standard *SOS* distress signal onto the AC power channel in Morse code by alternately loading and unloading one of the two cores in our MacBook. The resulting plot (Figure 8) clearly shows the distress signal.

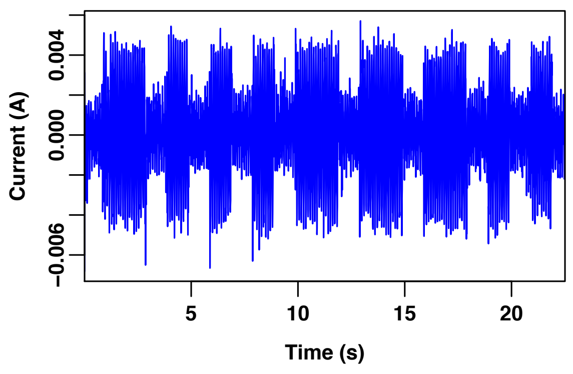


Figure 8: An excerpt of the *SOS* sent on the AC power covert channel. *SOS* is a repeated series of 3 *dits* followed by 3 *dahs*. In this case, the dits and inter-symbol timing are both 1 second long and the dahs are 2 seconds.

Acknowledgments

We thank the members of the SPQR group and Dina Katabi for discussions and feedback on drafts. We also thank George Danezis and the anonymous reviewers for Oakland 2012 for feedback on earlier versions of this work. This material is based upon work supported by the National Science Foundation under grants CNS-0923313, CNS-0845874, and two Graduate Research Fellowships. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was funded by a Sloan Research Fellowship.

References

[1] GPU Accelerated Compositing in Chrome. `http://www.chromium.org/developers/design-documents/gpu-accelerated-compositing-in-chrome`, Loaded February 10, 2012.

[2] Amazon Silk FAQs. `http://www.amazon.com/gp/help/customer/display.html/ref=hp_silk_faqs?nodeId=200775440`, Loaded February 2012.

[3] OnLive Game Service. `http://www.onlive.com/service`, Loaded February 2012.

[4] iMacros for Firefox. `http://www.iopus.com/imacros/firefox/`, Loaded September 2011.

[5] AGGARWAL, G., BURSZTEIN, E., JACKSON, C., AND BONEH, D. An analysis of private browsing modes in modern browsers. In *Proc. USENIX Security Symposium* (August 2010).

[6] ALEXA INTERNET, INC. Top 1,000,000 Sites (Updated Daily). `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`, Loaded February 12, 2012.

[7] ALEXA INTERNET, INC. Alexa top 500 global sites. `http://alexa.com/topsites/global`, Loaded September 2011.

[8] AUTHORS ANONYMIZED FOR SUBMISSION. Details anonymized for submission using the format specified by the PC Chair). Tech. rep., July 2011.

[9] BARISANI, A., AND BIANCO, D. Sniffing keystrokes with lasers/voltmeters. CanSecWest Presentation, March 2009.

[10] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] DAGON, D. DNS security: Lessons learned and the road ahead. Invited talk, USENIX Security Symposium, August 2009.

[12] ENEV, M., GUPTA, S., KOHNO, T., AND PATEL, S. Televisions, Video Privacy, and Powerline Electromagnetic Interference. In *Proc. ACM Conference on Computer and Communications Security (CCS)* (October 2011).

[13] FEDERAL COMMUNICATIONS COMMISSION. Code of Federal Regulations, Title 47, Part 15, Sections 101–103, October 2010.

[14] FREEDMAN, M. J., FREUDENTHAL, E., AND MAZIÈRES, D. Democratizing content publication with Coral. In *Proc. USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)* (March 2004).

[15] GUPTA, S., REYNOLDS, M. S., AND PATEL, S. 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)* (September 2010).

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

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

[18] HINTZ, A. Fingerprinting websites using traffic analysis. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)* (April 2002), R. Dingledine and P. Syverson, Eds., Springer-Verlag, LNCS 2482.

[19] HSU, C.-W. Multi-label classification. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#multi_label_classification`, November 2011.

[20] KASPER, T., OSWALD, D., AND PAAR, C. EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In *Workshop on Information Security Applications (WISA)*. August 2009.

[21] KIM, H., SMITH, J., AND SHIN, K. G. Detecting energy-greedy anomalies and mobile malware variants. In *Proc. International Conference on Mobile Systems, Applications, and Services (MobiSys)* (June 2008).

[22] KOCHER, P. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology (CRYPTO)* (August 1996).

[23] KOCHER, P., JAFFE, J., AND JUN, B. Differential power analysis. In *Advances in Cryptology (CRYPTO)* (August 1999).

[24] KRISHNAN, S., AND MONROSE, F. DNS prefetching and its privacy implications: When good things go bad. In *Proc. USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)* (April 2010).

[25] KUHN, M. G. Electromagnetic Eavesdropping Risks of Flat-Panel Displays. In *Workshop on Privacy Enhancing Technologies* (May 2004).

[26] KUHN, M. G. Security Limits for Compromising Emanations. In *Cryptographic Hardware and Embedded Systems (CHES)* (August 2005).

[27] KUHN, M. G., AND ANDERSON, R. J. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In *Information Hiding* (April 1998).

[28] LIBERATORE, M., AND LEVINE, B. N. Inferring the source of encrypted HTTP connections. In *Proc. ACM Conference on Computer and Communications Security (CCS)* (October 2006).

[29] LIU, L., YAN, G., ZHANG, X., AND CHEN, S. Virusmeter: Preventing your cellphone from spies. In *Recent Advances in Intrusion Detection*, vol. 5758 of *Lecture Notes in Computer Science*. 2009, pp. 244–264.

[30] LU, L., CHANG, E. C., AND CHAN, M. C. Website fingerprinting and identification using ordered feature sequences. In *Proceedings of the 15th European conference on Research in computer security* (Berlin, Heidelberg, 2010), ESORICS'10, Springer-Verlag, pp. 199–214.

[31] MOLINA-MARKHAM, A., SHENOY, P., FU, K., CECCHET, E., AND IRWIN, D. Private memoirs of a smart meter. In *ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys)* (November 2010).

[32] MURDOCH, S. J., DRIMER, S., ANDERSON, R., AND BOND, M. Chip and PIN is broken. In *Proc. IEEE Symposium on Security and Privacy (SP)* (May 2010).

[33] P3 INTERNATIONAL. P3 — Kill A Watt. `http://www.p3international.com/products/special/P4400/P4400-CE.html`, Loaded February 13, 2012.

[34] PATEL, S. N., ROBERTSON, T., KIENTZ, J. A., REYNOLDS, M. S., AND ABOWD, G. D. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *Ubicomp* (2007).

[35] SONG, D. X., WAGNER, D., AND TIAN, X. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proc. USENIX Security Symposium* (August 2001).

[36] SUN, Q., SIMON, D. R., WANG, Y.-M., RUSSELL, W., PADMANABHAN, V. N., AND QIU, L. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy* (Berkeley, California, May 2002).

[37] 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.

[38] VAN ECK, W. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security 4* (December 1985), 269–286.

[39] VUAGNOUX, M., AND PASINI, S. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *Proc. USENIX Security Symposium* (August 2009).

[40] W3C HTML WORKING GROUP. 4.8.11 The canvas element. `http://www.w3.org/TR/html5/the-canvas-element.html`, Loaded February 10, 2012.

[41] WHITE, A., MATTHEWS, A., SNOW, K., AND MONROSE, F. Phonotactic reconstruction of encrypted VoIP conversations: Hookt on Fon-iks. In *Proc. IEEE Symposium on Security and Privacy (SP)* (May 2011).

[42] WRIGHT, C. V., BALLARD, L., MONROSE, F., AND MASSON, G. M. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *Proceedings of 16th USENIX Security Symposium* (Berkeley, CA, USA, 2007), USENIX Association, pp. 4:1–4:12.

[43] YEE, B., SEHR, D., DARDYK, G., CHEN, J. B., MUTH, R., ORMANDY, T., OKASAKA, S., NARULA, N., AND FULLAGAR, N. Native client: A sandbox for portable, untrusted x86 native code. In *IEEE Symposium on Security and Privacy* (2009), IEEE Computer Society, pp. 79–93.